WRDC-TR-90-8007
Volume V
Part 7

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 7 - Neutral Data Definition Language (NDDL) User's Manual

J. Althoff, M. Apicella, M. Bernier

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UM 620341100	5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. V, Part 7
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Service 6b. OFFICE SYMBOL *(if applicable)*	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI
6c. ADDRESS *(City,State, and ZIP Code)* 6970 Presidential Drive Fairborn, OH 45324-6209	7b. ADDRESS *(City, State, and ZIP Code)* WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Developmental Center, Air Force Systems Command, USAF 8b. OFFICE SYMBOL *(if applicable)* WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464
8c. ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB, Ohio 45433-6533	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
78011F	595600	F95600	20950607

11. TITLE *(Include Security Classification)*
See block 19 er's Guide

12. PERSONAL AUTHOR(S)
Control Data Corporation: Althoff, J. L., Apicella, M. L., Bernier, M. P.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr.,Mo.,Day)*	15. PAGE COUNT
Final Report	4/1 87-12/31/90	1990 September 30	235

16. SUPPLEMENTARY NOTATION

WRDC/MTI Project Priority 6203

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify block no.)*
FIELD	GROUP	SUB GR.	
1308	0905		

19. ABSTRACT *(Continue on reverse if necessary and identify block number)*

This document explains how to use the Neutral Data Definition Language (NDDL). The NDDL is used to store and maintain the schemas and mappings in the Common Data Model database. This manual also explains the syntax and semantics of each NDDL command.

BLOCK 11:

INTEGRATED INFORMATION SUPPORT SYSTEM
Vol V - Common Data Model Subsystem

Part 7 - Neutral Data Definition Language (NDDL) User's Manual

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NO. *(Include Area Code)* (513) 255-7371	22c. OFFICE SYMBOL WRDC/MTI

EDITION OF 1 JAN 73 IS OBSOLETE

DD FORM 1473, 83 APR

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.

Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface,and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

Table of Contents

Table of Contents

Table of Contents

List of Illustrations

SECTION 1

INTRODUCTION

The Neutral Data Definition Language, hereafter NDDL, is an interpretive language that was developed to populate and maintain the Common Data Model database (CDM1 Doc. Control No. CCS1 DS 620341000). As the NDDL is intended to be used by data processing professionals who are experts with various database management systems, a discussion of data- base management systems is not included in this document (see CDM Administrators Manual Doc. Control No. UM 620341000). Furthermore, an understanding of the CDM database is necessary.

The CDM Database uses a Three-Schema approach based upon "The ANSI/X3/SPARC DBMS Framework: Report of the Study Group on Data Base Management Systems". In the IISS, the Three-Schema Architecture is implemented through the CDM facilities to store each of the three types of schemas and the interschema mappings. The NDDL supports the population and maintenance of appropriate representations for each of the three types of schemas.

The conceptual schema is represented by an IDEF-1 model. The CDM stores this model in terms of entity classes, attribute classes, and relation classes.

The external schemas are represented by tables. The mappings between these tables and the IDEF-1 model of the conceptual schema are part of the CDM database.

The internal schemas are represented in terms of physical database components, including record types and inter-record relationships.

Sections 1 and 2 of this manual discuss how to interface with the NDDL syntax which is similar to that of the Neutral Data Manipulator Language (see NDML Guide PRM 620341200).

Section 3 explains the syntax and semantics of the NDDL commands.

Section 4 contains a list of NDDL reserved words.

Section 5 discusses the functionality of the NDDL modeling commands in detail, and contains corresponding examples.

Section 6 is an aid to mapping between the three schemas.

Section 7 is an aid to displaying the entire contents of the CDM.

Section 8 contains tables of datatypes that are supported by the languages used by NDDL.

Section 9 contains three appendices:

Appendix A: NDDL Error Listing

Appendix B: A glossary of the more important terms used in this guide.

Appendix C: A list of references.

1 NDDL Environment

NDDL can be executed from three environments within IISS:

1. batch environment
2. interactive environment without the IISS User Interface/Virtual Terminal Interface (UI/VTI)
3. interactive environment with the IISS User Interface/Virtual Terminal Interface (UI/VTI)

NDDL is run by executing the command procedure NDDL.COM. The environment is determined by the parameter, or absence of a parameter, to NDDL.COM.

1. Batch Environment

```
$ @NDDL filename
```

filename - redirect the input for NDDL to come from the file 'filename'.DAT

- redirect the output for NDDL to be written to the file 'filename'.OUT

- 'filename' is only the name of the file. No file extension(.DAT) is needed.

The user will also be prompted for the CDM username and password.

Example

```
$ @NDDL TEST1

CDM    CDM
```

TEST1.DAT is a file containing NDDL commands.

Note, this file name must follow the UNIX standard for naming files, not that of the host operating system.

2. Interactive Environment (without IISS UI/VTI)

$ @NDDL

CDM CDM

The NDDL "NEXT COMMAND" prompt will be displayed, at which time the user may enter any valid NDDL command.

Example

$ @NDDL

NEXT COMMAND--

create model TEST_MODEL;

--COMMAND SUCCFSSFUL, COMMIT PERFORMED--

NEXT COMMAND--

halt;

--COMMAND SUCCESSFUL, COMMIT PERFORMED--

3. Interactive Environment (with IISS UI/VTI)

$ VT100

This command activates the IISS UIMS interface. This will bring up a UIMS form which will prompt the user for USER ID, password and role. The user then enters the CDM which he will access into the USER ID, and enter the password into the password and role slots.

After entering this information, another form will come up. Under function, enter NDDL and hit the enter key. The other two fields on this form need not be entered.

The NDDL form will be displayed, at which time the user may enter NDDL commands. A description of the form and how it is used for NDDL is contained in the next section of this document "Using the NDDL Form".

Using the NDDL Form

This subsection describes the use of the NDDL form. The form manipulations are supported by the IISS User Interface/Virtual Terminal Interface (UI/VTI) software. This section will also describe the detailed forms procedures unique to the NDDL application. For general information regarding forms use, refer to the IISS UI/VTI Terminal Operator's Guide.

NDDL Form Description

The NDDL form, depicted in figure 1, appears immediately following NDDL activation with the -I parameter as explained in the "NDDL Environment" section. A description of the individual fields follows.

Field 1 - Current Model Field

This field displays the current model name. The current model field is modified each time a CREATE MODEL or ALTER MODEL command is successfully executed. If neither command has been executed in a session, the "NOT YET SPECIFIED" message will appear in this field.

Field 2 - Current Database Field

This field displays the current database name. The current database field is modified each time a DEFINE DATABASE, ALTER DATABASE, DEFINE SET or DEFINE RECORD command is successfully executed. If none of the commands have been executed in a session, the "NOT YET SPECIFIED" message will appear in this field.

Field 3 - Current Output Mode Field

This field displays the current output mode, whether to a file or screen. This field is modified each time a SET OUTPUT command is successfully executed. If the command has not been executed in a session, the "NOT YET SPECIFIED" message will appear on the screen.

Field 4 - Current Automatic/Manual Commit Field

This field displays the current commit mode, whether commit is automatic or manual. This field is modified each time a SET COMMIT command has been successfully executed. If the command has not successfully executed, the "NOT YET SPECIFIED" will appear in this field.

Field 5 - NDDL Command Field

This is the field in which NDDL commands are entered. In addition, any generated output from the NDDL commands will be displayed in this field. For example:

- All "Copy" Commands
- Compare Model
- Check Model
- Merge Model, etc.

This field is 78 columns wide and 100 lines long. This field may be paged and scrolled as described in the IISS UI/VTI Terminal Operator's Guide.

Field 6 - Message Line Number Field

The message line number field serves two purposes. It is used to display the number of the message displayed in field 6, the message line. The message line number is also used to select a particular message in field 6 for viewing. The user can position the cursor to this field, enter a number from 01 to 99 and read the corresponding message, assuming one exists.

Field 7 - Message Line

The message line is 99 lines long, viewed one line at a time. This field may contain informative and error messages. In addition, this field will contain summary messages indicating whether or not errors have occurred and the number of generated messages. Use the <MESSAGE QUEUE> key to display the entire message queue. Position the cursor anywhere within Field 5 and press <PF3>.

Form Use

VT100 Keypad

The VT100 keypad is depicted in figure 2. The keys which apply uniquely to NDDL are:

pf17 - Press pf17 to terminate NDDL. This key forces an NDDL HALT command, freeing the user from explicitly entering the HALT.

enter - Press enter to initiate execution of the entered NDDL command(s). After the commands are executed or syntax checked, the form reappears. If an error was encountered, all commands reappear along with their error messages. If, however, no errors are encountered, the commands do not reappear when the form is repainted.

0 (pf16)- Press 0 to initiate execution of the entered NDDL command(s). After the commands are executed or syntax checked, the form reappears with the previously entered commands showing whether or not errors occurred. This feature is intended to be used when the user has a number of similar commands to execute. The user need only modify that dynamic portion of the command for each subsequent execution.

All other keypad keys operate as described in the UI/VTI Terminal Operator's Guide. In addition, there are a number of non-keypad keys which manipulate the form and which are described in the same document.

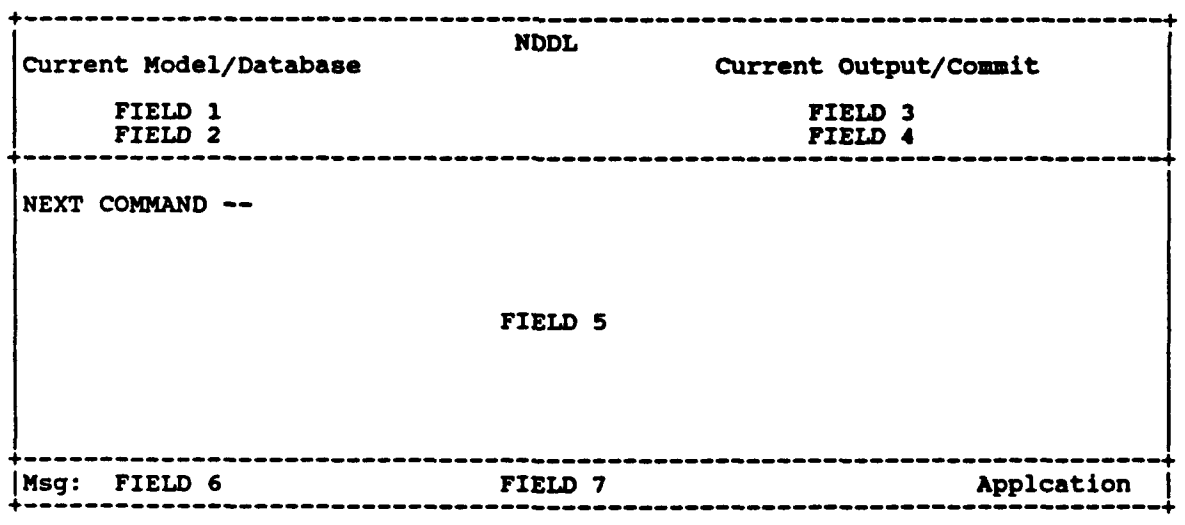


Figure 1: NDDL Form

pf1 GOLD	pf2 help	pf3 debug message queue	pf4 quit
7 (pf5) scroll left/ save command	8 (pf6) scroll right	9 (pf7) scroll up	- (pf8) scroll down
4 (pf9) page left insert	5 (pf10) page right delete line	6 (pf11) page up	, (pf12) page down
1(pf13)	2 (pf14)	3 (pf15)	enter Execute NDDL commands - redisplay the commands only if errors encountered
0 (pf16) Execute NDDL commands redisplay the commands unconditionally		. (pf17) HALT	

Figure 2: VT100 Function Keypad

NDDL Command Entry

All NDDL commands must be entered in the NDDL command field with syntax as described in the "NDDL Commands" section of this document. The terminating semicolons are mandatory for all commands. The user may enter as many complete commands as may be contained in 100 lines. After execution has started, the form will not reappear until all commands have executed or had their
errors diagnosed.

Error Message Reporting

After a form of commands has executed, error messages may or may not appear. If no errors have occurred, a "no errors encountered" message will appear in the message line of the form. If errors have occurred, a "command(s) completed - n error message" message will appear with n equal to the number of messages generated. Note that a particular command may generate many error messages. The

message area may contain up to 99 messages. Position the cursor to the message line number field (field 4), key in 01 and press enter for the first message. For each subsequent message, increment field 4 by one and press enter. To view the entire message queue, key in <pf3>. To return to the original form, key in <pf4>.

If many commands have been entered and errors were encountered, all commands preceding the erroneous command will have executed, applying their expected database modifications. The erroneous command and all subsequent commands in the same form will not execute. The commands following the erroneous command on the same form will be syntax checked only.

To assist in matching the error messages with the proper NDDL commands, the following messages are generated:

--COMMAND SUCCESSFUL COMMIT PERFORMED --	- The command has successfully executed.
--NO CHANGE MADE--	- Errors were encountered. The detailed error messages which apply to this message precede it.
--COMMAND SYNTAX CORRECT--	- An error in a previous NDDL command caused all subsequent commands in the form to be syntax checked only.
--INVALID SYNTAX--->	- The command syntax is incorrect.

As an example, if 5 "--COMMAND SUCCESSFUL--" messages are generated before the first error message appears, the offending NDDL command is the 6th one on the form.

SECTION 2

NDDL COMMAND SYNTAX

In the syntactic description of the NDDL commands, the following symbols are used:

[ ]	indicates an optional word or phrase
{ }	indicates a choice of only one word or phrase
...	indicates repetition of the last element
;	indicates the end of a command and must be entered by the user
|	indicates a choice of options

All lower case words must be specified as indicated; UPPERCASE words indicate a user-defined variable which the user may fill in with upper or lower case letters. Upper and lower case may be used anywhere. UPPERCASE words are case sensitive, i.e., XYZ is not the same as xyz. The user may enter any string of up to 30 characters for UPPERCASE words. These characters may be any combination of letter, digit, dash or underscore.

There is a maximum limit of 25 tokens (user defined variables) that may be placed on a parser list.

For example, the user may define a record with only 25 fields in one command. If the user chooses to specify more than 25 fields, he may Alter record and add the additional fields, again specifying only 25 fields each time.

Comments may be embedded by surrounding the comment with /* and */.

Most repetitions are separated by a single blank space; however, a few commands must be separated by commas.

SECTION 3

NDDL COMMAND FORMATS

This section describes each NDDL command. The syntax of each command is given and followed by command semantics and command examples.

ALTER ALIAS

Syntax:

```
alter alias  { entity EC_NAME1     [is] EC_NAME2
             | attribute AC_NAME1 [is] AC_NAME2  };
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   entity or attribute class with both primary and alias names

2. EC_NAME1 and AC_NAME1 are the current primary names; EC_NAME2 and AC_NAME2 are the current aliases.

3. The command switches the current primary and the alias names and vice-versa.

Examples:

```
alter alias attribute ORDER_NUMBER  ORDNO ;
alter alias entity CUSTOMER is CUST ;
```

ALTER ATTRIBUTE

Syntax:

```
alter attribute [class] AC_NAME [domain DOMAIN_NAME]
    [drop keyword KEYWORD...]
    [add keyword KEYWORD...]
      [[ownership to entity [class] EC_NAME
      [as {member of key [class] KC_NAME [migrates thru
                                    RC_NAME]...}
          {nonkey                                  }]];
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   model
   attribute

   alter attribute [class] AC_NAME

1. An attribute is altered with one of the options specified.

   [domain DOMAIN_NAME]

1. When a DOMAIN_NAME is specified the validity of the Domain is checked and processing will halt if it is invalid.

2. If the attribute is a discriminating attribute,the new domain must still contain the specific values required by the category relation already established. Else, no domain change will occur for this attribute.

3. If the DOMAIN_NAME is valid ATTRIBUTE_CLASS will be updated to indicate the new DOMAIN_NO.

   [drop KEYWORD...]

1. Keyword references are deleted from the attribute being altered.

   [add KEYWORD...]

1. The keyword is created if it does not already exist.

2. The attribute keyword reference is created.

[Ownership to entity [class] EC_NAME
[as {member of key [class] KC_NAME [migrates thru RC_NAME]...}
    {nonkey   }]]

1. The following elements must exist in the Common Data Model:

- o Attribute whose ownership is to be altered
- o Entity that already owns the attribute
- o Entity to be the new owner, that is different than the old owner
- o Link Relation thru which the key is to be migrated

2. The ownership of the attribute is altered from the old owner to the new owner.

   a) Ownership of category discriminating attribute is allowed to migrate upward from the generic entity to outside the category structure through a complete link relation only.

3. If the clause "AS MEMBER OF KEY...MIGRATES THRU" is specified, the attribute can not be added to a primary key involving either the old or new owner entity in a category relation. Keys used in category relations cannot be modified.

4. If the clause "AS MEMBER OF KEY" is specified, the attribute is placed as a key member in key KC_NAME. If KC_NAME did not previously exist, a new key occurrence is created for that entity. This is added as a primary key if a primary key does not exist. Else, it will be an alternate key.

5. If the clause "MIGRATES THRU" is specified on the "AS MEMBER OF KEY" clause, there must exist a one to one correspondence between KC_NAME and RC_NAME; i.e., the link relation name specified as RC_NAME must be the relation migrated by key KC_NAME. This clause is used for two semantic purposes.

   a. When a new key, KC_NAME is specified and RC_NAME is an incomplete relation, and the old owner entity is directly dependent on the new owner entity, the link relation will be made complete; i.e., KC_NAME will be re-migrated to the old owner entity, and all previous migrations of the attribute will be preserved.

   b. When more than one link relation exists between the old owner and new owner migrating the same existing key KC_NAME, the link relation RC_NAME will be chosen to migrate key KC_NAME back to the old owner, with the attribute's role name unchanged. All other link relations would migrate the attribute and modify its role name. If the user did not specify RC_NAME for this case, the computer would arbitrarily choose any link relation for the unchanged role name.

6. If the attribute can migrate from the new owner back to the old owner; i.e., if conditions 5a or 5b are present, then the attribute in the old owner is converted from owned to inherited and it continues to belong to any keys that it did previously.

a.
- the old owner is dependent on the new owner in a link relation
- a key has migrated through that link relation
- the attribute is being placed in that key in the new owner
- and that key specified as KC_NAME previously existed

b.
- the old owner is dependent on the new owner in a link relation
- that link relation is specified as RC_NAME in the "migrates thru" clause, a key has not migrated thru that link relation
- and the attribute is being placed in that key in the new owner.

7. If the attribute cannot migrate from the new owner back to the old owner; i.e., if any of the following conditions are present, then the attribute is dropped from the old owner and any attributes in other entities that were dependent on it are also dropped.

   - The old owner is not dependent on the new owner in any link relations.
   - Or the old owner is dependent on the new owner, but no key has previously migrated through the link relation (and a key/relation has not been specified thru which the attribute can migrate).
   - Or a key has migrated through the link relation, but the attribute is not being placed in that key.
   - Or a new key is specified but the link relation necessary to migrate the key is not specified on the "migrates thru" clause.

8. If the attribute has not previously migrated from the old owner into the new owner; i.e., if it does not already exist as an inherited attribute in the new owner, it is added to the new owner. In addition:

   - All inherited attribute use classes of the attribute being moved will be lost.
   - If AS MEMBER OF KEY is specified.

     Migration as a primary key member is not allowed if there are category relations involving the old and new owner entities. Keys used in category relations cannot be modified.

     If the keys listed are already in existence, the attribute is placed in each of listed keys and new inherited attributes are added to the entities that those keys have migrated to. These new inherited attributes are created as nonkeys.

If the keys listed do not exist, a new key is created and the attribute is placed in the new key. The new key will be a primary key if a primary key does not exist. Else, it will be an alternate key. In addition, if the "MIGRATES THRU" relation clause is specified, new inherited attribute use occurrences are created for each entity in the migration path of the attribute.

- o If AS NONKEY is specified or if no AS clause is specified, the attribute is not placed in any of the new owner's keys.

9. If the attribute has previously migrated from the old owner into the new owner; i.e., if it already exists as an inherited attribute in the new owner, it is converted from inherited to owned. In addition:

   - o Inherited attributes of the original attribute use will be lost for all migrations to dependent entities, except where the dependent entity is the new owner entity.

   - o If AS MEMBER OF KEY is specified.

   Migration as a primary key member is not allowed if there are category relations involving the old and new owner entities. Keys used in category relations cannot be modified.

   If the keys listed are already in existence, the attribute is placed in any of the listed keys that it did not previously belong to, and new inherited attributes are added as nonkeys to the entities that those keys have migrated to.

   If the keys listed do not exist, a new key is created and the attribute is placed in the new key. The new key will be a primary key if a primary key does not exist. Else, it will be an alternate key. In addition, if the "MIGRATES THRU" relation clause is specified, new inherited attribute occurrences are created for each entity in the migration path of the attribute.

   - o If AS NONKEY is specified, the attribute is removed from any of the new owner's keys that it previously belonged to, and any attributes in other entities that were inherited from it are dropped.

   - o If no AS clause is specified, the attribute continues to belong to any keys that it did previously in the new owner and the new owner's dependent entity. Use this clause if migrations of the new owner entity are to be preserved.

10. If the attribute has previously migrated more than once from the old owner into the new owner; i.e., if it already exists as two or more inherited attributes in the new owner, only one of those inherited attributes may belong to any of the new owner's keys. If more than one does, the modeler must remove the extras from the keys before altering the ownership of the attribute.

This restriction is necessary because an owned attribute can appear only once in its owner entity. Thus, the NDDL Processor would have to select one of the inherited attributes in the new owner, convert it to owned, and drop the rest. This would involve dropping key members if more than one of the inherited attributes belonged to keys in the new owner. To avoid having the NDDL Processor decide which key members to drop and which to retain, the modeler is required to make that decision before altering the ownership of the attribute.

11. If the attributes migrates more than once from the new owner into a dependent entity; i.e., if all the following conditions are present:

    - o The same entity is dependent on the new owner in two or more relations
    - o Keys have migrated through those link relations (not necessarily the same one through each)
    - o And the attribute is being placed in those keys in the new owner, then multiple inherited attributes are created in the dependent entity, one per link relation. The NDDL Processor will generate unique names for these inherited attributes by offsetting a number with the circumflex character to the name of the attribute. The modeler can then rename them by using ALTER ENTITY command.

12. In some of the situations described above, an owned or inherited attribute is removed from a key. If the key contains other members, it is assumed to still be valid, but a warning message is issued that it is now suspect because it has been modified. If, however, the attribute is the sole member of the key, the key itself is dropped. Also, if the key had migrated through any relation, the inherited key that resulted will be dropped, and will leave those relation incomplete.

13. The old and new owners must not be dependent on each other, either directly or indirectly; i.e., they must not be in a dependency loop. For example, if entity class ABC is dependent on entity XYZ through one series of link relations, XYZ must not be dependent on ABC through another series. The CHECK MODEL command can be used to find any such loops.

14. Attributes, either owned or inherited, that have mappings to external or internal schemas cannot be dropped. Therefore, ALTER ATTRIBUTE commands that involve dropping such attribute based on any of the situations described above are rejected.

Examples:

```
alter attribute AC_ORDER_INFO domain ADDRESS
    add keyword ZIP
    drop keyword COUNTY;
```

```
alter attribute PART_INFO ownership
    to entity PART_ORDER
    as member of key PART_KEY_NEW
    migrates thru IS_ORDERED_BY;

alter attribute PART_STOCK_ID ownership
    to entity PART_ORDER
    as nonkey;

alter attribute COUNTRY ownership
    to entity ADDR
    as member of key ADDR_KEY;
```

ALTER CATEGORY

Syntax:

```
alter category [relation] RC_NAME
      of EC_NMAE [to(complete)]

[add
  ([category] EC_NMAE1 if 'literal'}...
   [keyword KEYWORD...]]
[drop
  ([category] EC_NAME1}...
   [keyword KEYWORD...]];
```

Semantics:

1. To complete the above command, the following elements must exist in the Common Data Model:

   model
   category relation
   generic entity (EC_NAME)
   category entity (EC_NAME1)

```
alter category [relation] RC_NAME
    of EC_NAME [to (complete  )]
               (incomplete)
```

1. The categorization may be changed from complete to incomplete or incomplete to complete.

2. A complete category relation requires at least two category members.

```
[add
   ([category] EC_NAME1 of 'literal'}...
   [keyword KEYWORD...]]
```

1. The category relation is adjusted by adding entities.

2. The primary key of the generic entity will be migrated as the primary key of the category entities. A category entity cannot have any keys before becoming part of the category relation.

3. The literal value must be unique for each of the category entities in the category relation.

4. The literal value must be a valid specific value in the domain of the discriminator.

5. The keyword is created in the Common Data Model if it doesn't already exist.

6. A relation keyword reference is also created.

```
[drop
   ([category] EC_NAME1}...
   [keyword KEYWORD...]];
```

1. The category relation is adjusted by deleting entities.

2. The key migration will be dropped from each relation and entity dependent upon the category entity being dropped from the model.

3. The relation keyword reference is deleted.

Examples:

```
alter category relation EMPLOYEE_TYPES
    of EMPLOYEE to complete
    add category FULLTIME_EMP if "F"
    drop keyword EMPL;
```

ALTER DATABASE

Syntax:

```
alter  {pcb     }
       {database}   DB_NAME        [to host    HOST_ID]
                [with password  PW_ID]
                [schema  S_NAME and subschema SS_NAME]
                [located at 'STRING']
                [position INTEGER1 in psb PSB_NAME
                      feedback length INTEGER2]
           [[stores] character null [as] {zeros        }
                                         {spaces       } ]
                                         {null         }
                                         {all 'STRING2'}

           [[stores] integer null [as]   {zeros        }
                                         {low-values   }
                                         {high-values  }
                                         {null         }
              [ntm directory 'STRING3']

              [add areas  {AREA_NAME}...]
              [drop areas {AREA_NAME}...]
```

Semantics:

1. The user will not be able to change the DBMS of a database.

2. The user will be able to change any of the following information about the database:

   host
   password
   schemas
   location
   areas
   PBS information
   NTM directory
   null values

3. The user will be able to add and drop areas associated with the database.

4. The user may not drop the last area associated with the database.

5. The position clause (for IMS) must be used in its entirety even if only one or two of the three parameters change.

6. The alter database command establishes the current database.

7. NTM directory and null value information can be altered for any database.

ORACLE Database

```
alter database DB_NAME  [ to host HOST_ID ]
                        [ with password PW_ID ];
```

1. A password may be altered.

2. The host_id may be altered.

VAX-11/IDMS/IS-II database

```
alter database DB_NAME  [ to host HOST_ID ]
   [schema S_NAME and subschema SS_NAME ]
   [located at 'STRING']
   [add  (areas  AREA_NAME...)]
   [drop (areas  AREA_NAME...)] ;
```

1. Host_id may be altered.

2. Schema or subschema may be altered.

3. Location where a VAX-11 database is stored may be changed.

4. Areas may be added and/or dropped.

IMS Database

```
alter pcb DB_NAME [ to host HOST_ID]
[ position INTEGER1 in PSB_NAME
     feedback length INTEGER2 ] ;
```

1. The HOST_ID may be altered.

2. The position clause must be completely used, even if only one or two of the three parameters change.

Examples:

```
alter database ORC_DB to host VAX with password ORC_PW ;

alter database COD_DB add areas A3 drop areas A1 ;

alter pcb IMS_DB with position 1 in psb SS_PSB
     feedback length 30 ;
```

Syntax:

```
alter dbms DBMS_NAME   [model DBMS_TYPE]
       [add host  HOST_ID...]
       [drop host  HOST_ID...]  ;
```

Semantics:

1. DBMS_TYPE must be one of the following:

   H - hierarchic
   R - relational
   N - network
   I - indexed
   S - sequential

   This clause is used if a change in dbms model type is needed.

2. This command can also be used to add and remove hosts associated with a particular dbms. This is the same cross reference maintained by ALTER HOST command.

3. When dropping a host association, no database may be defined for that host on that dbms.

4. Adds will be done before drops.

Example:

```
alter dbms S2K type H
    add host  CYBER;

alter dbms TOTAL
    drop host IBM CYBER;
```

ALTER DOMAIN

Syntax:

```
alter domain DOMAIN_NAME
  [add  [([data] type DATA_TYPE_NAME nddl_data_type integer1
[:integer2])...]
            [value 'SPECIFIC_VALUE'...]
            [range ("BEGIN_VALUE' thru
'ENDING_VALUE')...|all]]
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   domain
   data type

2. The data type being processed is updated with the altered information.

3. In specifying the maximum size of a data type, Integer2 cannot be greater than Integer1.

4. See Section 8 for valid NDDL datatypes.

```
add ([data] type DATA_TYPE_NAME nddl_data_type integer1
     [:integer2])....
     [value 'SPECIFIC_VALUE'...]
     [range ('BEGIN_VALUE' thru 'ENDING_VALUE')...]
```

1. A data type is added that is specified in this clause.

2. The data type will be inserted as user defined.

3. The value or values specified will be added.

4. The range or ranges specified will be added.

5. Values and ranges are associated with the standard datatype.

```
drop ([data] type DATA_TYPE_NAME)...
      [value ('SPECIFIC_VALUE')...|all]
      [range ('BEGIN_VALUE' thru 'ENDING_VALUE')...|all]
```

1. If it is determined that the data type is still associated with a data field, data item or attribute the data type cannot be dropped. All attributes that use this data type being dropped are reported to the user.

2. A standard data type cannot be dropped.

3. The value or values specified will be deleted.

4. If the "value all" clause is selected, all values will be deleted for the domain.

5. The range or ranges specified will be deleted.

6. If the "range all" clause is selected, all ranges will be deleted for the domain.

```
alter [data] type DATA_TYPE_NAME
      [nddl_data_type integer
      [:integer2]]
      [to standard]
```

1. If the data type is altered to "standard" it will become the current standard, and the previous standard will become a user defined data type for the same domain.

2. The data type may also be altered to another legal type with a new size and decimal specifications.

Examples:

```
alter domain ADDRESS
add data type ALPHA_NUMERIC integer 5:2
        VALUE 'ABCDE' 'A1B2C6STND'
drop data type NUMERIC
        value '88.55'  '76.32'
        range all
alter type ALPHA character 30 to standard;
```

ALTER ENTITY

Syntax:

```
alter entity [class] EC_NAME
  [add  [[(primary  )] [key [class] KC_NAME[= AC_NAME ...]]...
         (alternate)

         [[owned] attribute [class] AC_NAME ...]
                  [keyword KEYWORD ...]]
         [drop  [key [class] KC_NAME ] ...
                [[owned] attribute [class] AC_NAME ...]
                [keyword KEYWORD ...]];
         [alter [attribute [class] OLD_TAG_NAME to
                           NEW_TAG_NAME]...
                [key [class] KC_NAME [to (NEW_KC_NAME)]
                                         (priamry    )
                 [add attribute [class] ADD_AC_NAME...]
                 [drop attribute [class] DROP_AC_NAME ...]]
                 [substituting]    ];
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   entity
   attribute to be dropped, added or altered
   key to be dropped or altered

```
add key [(primary  ) [class] KC_NAME [=AC_NAME ...]]
        (alternate)
```

1. An occurrence of attribute for AC_NAME must exist

2. A new attribute use class occurrence is created for AC_NAME if one does not already exist. If the attribute use class does already exist, AC_NAME may not be owned by any other entity.

3. A new occurrence of key is created for the entity using KC_NAME.

4. A new occurrence of key member is created for the entity for each AC_NAME. If AC_NAME is omitted, a key member occurrence is created using KC_NAME as the name of the attribute.

5. The key will be a primary or alternate key as specified. If it is not specified, the key will be the primary key if a primary key does not exist. Else it will be an alternate key.

6. Only an alternate key class may be added to generic or category-member entity.

add [owned] attribute [class] AC_NAME ...

1. Required existence in the Common Data Model - attribute.
2. New attributes for the entity being created are added.
3. Each attribute is created as an owned attribute for the entity.
4. A new occurrence of the attribute as an attribute use class is created.

add keyword KEYWORD ...

1. Keyword references are created for the entity.
2. The keyword will be created in the Common Data Model if it does not already exist.

drop key [class] KC_NAME ...

1. The key for the entity is deleted.
2. All key members for the key are deleted.
3. If the key being dropped is from a complete relation, that complete relation is deleted.
4. All attribute use classes which were formed from a migration of that key member and any migration of those attribute use classes are deleted.
5. The inherited attribute use occurrence of a migrated attribute use class is deleted.
6. The inherited attribute will not be dropped, when that OAK is mapped to fields, sets, unions, partitions or algorithms.
7. This is allowed for an alternate key of an entity that is in a category structure, but disallowed for any primary key use in a category structure.

drop [owned] attribute [class] AC_NAME ...

1. The owned attribute occurrence and the attribute use class for each AC_NAME is deleted from the entity being altered.
2. The inherited attribute will not be dropped, when that AUC is mapped to fields, sets, unions, partitions or algorithms.

3. If the attribute is the discriminator for a category relation, the attribute will be deleted as well as the category relation with complete ripple down, ie. all subordinate category relations will be dropped.

drop keyword KEYWORD ...

The entity keyword reference is deleted.

alter [attribute [class] OLD_TAG_NAME to NEW_TAG_NAME] ...]

1. The attribute to be altered must be in the specified entity as either owned or inherited.
2. The old and new names of the attribute must be different.
3. The specified entity must not already contain another attribute with the new tag name.
4. The name of the tag or role name of the specified entity is changed.

alter [key [class] KC_NAME [to {NEW_KC_NAME}]
{primary }

1. The key to be altered must be in the specified entity class.
2. If the key name is to be altered to a new key name, the NEW_KC_NAME must not already belong to that entity.
3. If an attribute class is being added to the key, it must be in the specified entity as either owned or inherited, and it must not already belong to that key.
4. The attribute is added to the key, and if that key has migrated through any relations, inherited attributes are added as nonkeys in the dependent entities.
5. If an attribute is being dropped from the key, it must belong in that key, and it must not be the only member. If it is, the entire key must be dropped instead, using the Drop Key clause.
6. The attribute is dropped from the key, and any inherited attributes that resulted from the migration of that attribute are dropped. This is a recursive delete for every level of dependency of attribute DROP_AC_NAME.
7. The type of key may be altered from alternate to primary. If a primary key already exists, its type will be changed to alternate the specified key will become the new primary key.
8. There must always be a primary key.

```
alter key [class] KC_NAME
add attribute [class] ADD_AC_NAME...
drop attribute [class] DROP_AC_NAME...
substituting  ;
```

1. Use of this clause would require that a like number of attributes be added and dropped from the key at the same time. The key members specified will have positional significance; i.e., the first attribute in the add clause will be the substitute for the first attribute in the drop clause and so on.

2. The attribute being added and dropped must all belong to the entity being altered, as either owned or inherited.

3. The attributes being dropped must exist in the key.

4. The attributes being added must not previously belong in the key being altered. They may be key members of a different key in the entity being altered.

5. The new key members of key KC_NAME, listed as ADD_AC_NAME, are substituted for those listed as DROP_AC_NAME.

6. All inherited attributes of DROP_AC_NAME are updated to reflect that they are now inherited from ADD_AC_NAME, thus preserving existing migrations to all the dependent entities.

7. The role names of the attributes in each dependent entity in the migration path will be modified to reflect the user specified ADD_AC_NAME. If this role name already exists or any dependent entity, an error message is issued.

8. For primary keys migrated through category relations, add, drop and substituting of attributes is not allowed. Primary keys used in category relations cannot be modified.

Examples:

```
alter entity class EC_CUSTOMER
    add key class KC_CUST_INFO = AC_CUST_NAME
        keyword CUSTOMER;

alter entity EC_ORDER
      alter attribute ORDER_DATE to ORDER_START_DATE;

alter entity PARTS
      alter key PART_KEY
           add attribute SHIP_LENGTH SHIP_WIDTH
           drop attribute SHIP_HEIGHT SHIP_SIZE
           substituting;
```

Syntax:

```
alter  {field   }   FIELD_NAME  of  {table  }  REC_NAME
       {column  }                   {record }
       {element }                   {segment}
       {item    }

         [of     {database}    DB_NAME ]
                 {  pcb   }

  [  {data]   type    DATA_TYPE_NAME} ]
  {                        null    }

     [occurs  INTEGER1 [depending on     {FIELD_NAME_2} ]]
                                         {       null }

     [ {is not indexed         } ]
       {indexed by FIELD_NAME1}

     [redefines    {FIELD_NAME_3}]
                   {null}

     [ {unknown} ]
       {known   }

    [[ {unique    }
       {duplicate }  ]  key ] ;
       {not       }
```

Semantics:

1. This command can be used to change the characteristics of a data field.

2. The of database clause is used for identification only, the field cannot be changed to another record or data base.

3. The data type of the data field can be changed, or converted to null in the case of a group data field.

4. The depending on field can be added, changed or removed (the latter by using the null).

5. The redefines data field can be added, changed or removed (the latter by using the null).

6. A field can be changed to either known or unknown by the dbms. A field is unknown to its dbms if it does not appear in the schema or database definition for that database.

7. A field can be converted to key or nonkey or be changed to a unique key or a duplicates allowed key.

8. A field can be changed to a COBOL indexed by field or from an index to a non-index. A data type name, if not specified for an index, will default to a numeric index data type.

9. Changing the level and structure of components is not permitted. The record should be dropped and recreated or fields dropped and added with an ALTER RECORD command.

10. When changing the data type of a field, the new type must be a valid option for the related DBMS.

    Relational DBMS: ORACLE, DB2, INGRES5, INGRES6

    C- Character
    P- Packed
    I- Integer
    V- Variable length Character
    A- Small Integer

    CODASYL DBMS: IDMS, VAX-11

    C- Character
    P- Packed
    I- Integer
    F- Float
    V- Variable
    A- Small Integer
    O- Double Precision
    N- Unsigned

11. For ORACLE databases, data types of integer or small integer must have a size of 4 or 9 only.

Examples:

```
alter field FIELD_A of record REC1 type null; (to allow this
to be a component data field)

alter field FIELD_B of record REC1 type NEW_TYPE occurs 5
depending on null redefines BIG_GROUP;

alter field FIELD_C of record REC1 unknown;

alter field FIELD_D of record REC1 not key;
```

Syntax:

```
alter host HOST_ID

    [add dbms DBMS_NAME...]

    [drop dbms DBMS_NAME...] ;
```

Semantics:

1. This command shall be used to add and remove dbms associations with a particular host.

2. The HOST_ID and the DBMS_NAME mentioned must exist in the CDM.

3. To drop a dbms association, no databases may be defined for that dbms on that host.

4. Adds will be done before drops.

Examples:

```
alter host M5
          add dbms  INGRES RIM
          drop dbms S2K;

alter host F5
          add dbms RIM;

alter host X5 drop dbms ORACLE;
```

ALTER MAP

Syntax:

for entity:

```
alter map EC_NAME

          [add record  (DB_NAME.REC_NAME}...]
          [drop record (DB_NAME.REC_NAME}....]
          [(allow}     retrieval][(allow}    update];
          [(disallow}            ][(disallow}       ]|
```

for attribute:

```
alter map EC_NAME.TAG_NAME for preference PREF_NUMB

          [                    (replication    )]
          [                (relational     )]
          [(active )       (original source)]
          [(passive)           (copy           )]
          [                (redundant      )]
          [                (backup         )]

          [to preference PREF_NUMB2]

          [add [field (DB_NAME.REC_NAME.FIELD_NAME) ...]

               [set (DB_NAME.SET_NAME value 'STRING1')
...]]

          [alter set (DB_NAME.SET_NAME value 'STRING2')
...]

          [drop [field (DB_NAME.REC_NAME.FIELD_NAME) ...]

               [set (DB_NAME.SET_NAME)...]] ;
```

For link relation and category relation:

```
alter map EC_NAME1 RC_NAME [EC_NAME2]

          [add   set
(DB_NAME.SET_NAME[.MEMBER_REC_NAME])...]
          [drop  set
(DB_NAME.SET_NAME[.MEMBER_REC_NAME])...]    ;
```

Semantics:

1. PREF_NUMB is used to identify the existing mapping. If PREF_NUMB2 is specified the mapping preference will be changed. If a mapping for PREF_NUMB2 exists, it will be switched to PREF_NUMB. The 'to preference' clause cannot be used in conjunction with the add, alter, and drop clauses. Preferences 1 thru 50 specify ACTIVE mappings. Preferences 51 to 99 specify PASSIVE mappings.

2. There can only be one mapping from a tag to a data field for a stated preference. When an entity is horizontally partitioned, a tag may map to two or more data fields for the same preference. This command allows data fields to be added or dropped for the same reference, only if the entity is horizontally partitioned.

3. More set values mappings can be added.

4. Set value mappings can be dropped.

5. The alter clause is used to change the values for existing AUC to SET mappings. The value must not have been used before for this mapping unless the entity is horizontally partitioned, in which case the same set value may be used for a different fragment of the partition. In this clause DB_NAME and SET_NAME are used for identification.

6. The following rules apply when altering Link Relation Category Relation (RC) to set member mapping:

   a. ALTER ADD rules are the following:

      - The MEMBER_RECORD_NAME may be omitted if the to be mapped to is a single member record type set; otherwise the member record name is required.

      - There must be no previous mappings to the set by any other relations or AUC's.

   b. ALTER DROP requires a MEMBER_RECORD_NAME entry if, by omitting, it, the "to set" specification would be ambiguous.

   c. Link Relations require independent_name, RC_NAME and dependent_name. Category Relations require generic_name and re_name.

7. The user may specify any one of six mapping classifications for the type of mapping, whether ACTIVE or PASSIVE.

8. A mapping from an entity may add or drop record mappings.

9. The "allow" "disallow" clause for retrieval and update may be changed.

Examples:

```
alter map PART.PART_SIZE for preference 2
          to preference 3 ;
alter map ORDER.STATUS for preference 1
          add set CUSTDB.LOST value '999'
          alter set CUSTDB.INPROCESS value '777';
```

```
alter map EMPLOYEE
          drop record SKILLDB.EMPLOYEE
          add record PERSDB.PERSON;

alter map EMPLOYEE disallow update;

alter map PART.PART_SIZE passive backup
          for preference 51;

alter map PERSONNELL GENDER_TYPE
          ADD SET PEOPLE.PERSONS;
```

Syntax:

```
alter model MODEL_NAME;
```

Semantics:

```
alter model MODEL_NAME;
```

1. Required existence in the Common Data Model: model class.

2. The model is updated and marked as unchecked.

3. The model becomes the "current" model for all other modeling commands and remains current until another CREATE MODEL or ALTER MODEL command is entered.

Examples:

```
alter model ABC_COMPANY;
```

ALTER MODULE

Syntax:

```
alter module MOD_NAME [in LANGUAGE]

   [drop parameters  PARM_ID...]

   [add parameters ({after }  PARM_ID
                    {before}

         {PARM-ID2 [data] type DATA_TYPE_NAME} ...)...   ;
```

Semantics:

1. This command may be used to change the definition of the module's parameter list or the language.

2. To add parameters, the before or after is necessary to indicate the position where the new parameter appears in the list.

3. To allow change of a parameter's data type, one command that drops the parameter and adds it back with the correct data type can be used. Therefore, all drops will be processed before adds.

4. The module definition cannot be altered if the module already has mapping algorithms established for it.

Examples:

```
alter module X52 in FORTRAN;
alter module X52
      drop parameters PART-WEIGHT
      add parameters after IN-REC
            PART-WEIGHT data type STD_WEIGHT
            PART-HEIGHT      type STD_HEIGHT   ;
```

ALTER PARTITION

Syntax:

```
alter partition  [INTEGER_1] of entity EC_NAME
     ([drop record DB_NAME.REC_NAME...])
     ([add record  DB_NAME.REC_NAME...]) ;
```

Semantics:

1. Records named in the ADD and DROP clauses must already be defined in the CDM.
2. INTEGER_1 is used to identify the partition.
3. If INTEGER_1 is not specified, a "1" will be assumed.
4. After the partition is altered it must contain at least two records.
5. The same record type must not participate in more than one partition of the same entity class.

Examples:

```
alter partition 2 of entity ORDERS
       add record   DB3.ORDER_INFO  ;

alter partition of entity ORDERS
       drop record  DB3.ORD  ;
```

Syntax:

```
alter psb PSB_NAME to host  HOST_ID;
```

Semantics:

1. This command only applies to IMS databases.
2. The PSB_NAME must be previously defined to the CDM.
3. The HOST_ID must be previously defined.
4. The PSB will be associated with the named HOST_ID, the association will change.

Examples:

```
alter psb ABCD to host XYZ ;
```

ALTER RECORD

Syntax:

```
alter    {table  }   REC_NAME   of  {database}  DB_NAME
         {record }                   {   pcb  }
         {segment}

[add         [areas  AREA_NAME ...]

              {columns }    [ {after  } ]  [FIELD_NAME_1]]
         [    {fields  }      {before}
              {elements}
              {items   }

          {   [LEVEL_NO ]  { FIELD_NAME_5 }
                           {    filler    }

          [{[data]   type    DATA_TYPE_NAME } ]
           {         filler_size            }

          [ occurs  INTEGER1  [  depending on FIELD_NAME_3 ] ]

          [ indexed by FIELD_NAME_2 ]

          [ redefines FIELD_NAME_4 ]

          [  {known   }  ]
             {unknown }

          [[ {unique   }     ]   key ]      } ... ] ]
             {duplicate}

 [drop    [ areas  AREA_NAME ...]

          [  {columns }   FIELD_NAME ...] ] ;
             {fields  }
             {elements}
             {items   }
```

Semantics:

1. The of database clause is required for identification of the records database if a current database has not been established. The record cannot be changed to another database.

2. Areas in which the CODASYL record is stored can be added and dropped. If dropping areas, at least one AREA/RECORD association must exist for a CODASYL data base after all drops are completed.

3. Data fields of the record can be dropped. This is equivalent to using the drop field command; please consult its semantics.

4. Data fields can be added to the record structure. Refer to DEFINE RECORD for the semantics of the field definition.

5. When a field is dropped, all of its subcomponent fields will also be dropped.

6. If the field being dropped is indexed, the index is dropped.

7. To change the definition of an individual field the user has two choices - the field may be dropped and re-added or the user can use the ALTER FIELD command.

8. The field will not be dropped if there are any associated mappings.

9. The LEVEL_NO cannot be changed, it is only used for identification.

10. FIELD_NAME_5 is the actual field being dropped or added. FIELD_NAME_1 identifies where FIELD_NAME is to be positionally added. FIELD_NAME_2 is the indexed data field name. FIELD_NAME_3 is the equivalent of COBOL's depending on clause. FIELD_NAME_4 is COBOL's equivalent of the redefines clause. Field names 1 through 4 must be previously defined.

11. A data field cannot be added before or after a FILLER.

12. A filler can be added without redoing the entire record, but cannot be dropped.

13. When adding fields with data types, the fields data type must be a valid option for the related DBMS.

    Relational DBMS: ORACLE, DB2, INGRES5, INGRES6.

    C- Character
    P- Packed
    I- Integer
    V- Variable length Character
    A- Small Integer

    CODASYL DBMS: IDMS, VAX-11.

    C- Character
    P- Packed
    I- Integer
    F- Float
    V- Variable length Character
    A- Small Integer
    O- Double Precision
    N- Unsigned

14. For ORACLE, data types of Integer or Small Integer can only have a size of 4 or 9.

Examples:

Alter record X1 add areas A1 A2 drop areas A3 ;

The following example shows how to translate from a COBOL record translate.

```
01  REC6.

     03  PART-JOB-GROUP.
                05  PART-COUNT           PIC 99.
                05  PART-ID.
                07  PART-LOC             PIC X(10).
                07  PART-NAME            PIC X(30).
         05  PART-JOB OCCURS 7 TIMES DEPENDING ON PART-COUNT.
                07  PART-JOB-TIME        PIC 9(9).
                07  PART-JOB-PLACE       PIC X(30).
         05  JOB-STATUS                  PIC X(8).
         05  JOB-INFO REDEFINES JOB-STATUS.
                07  JOB-ID               PIC XX.
                07  JOB-DATE             PIC X(6).
     03  PART-CUST-GROUP OCCURS 100 TIMES INDEXED BY C-INDEX.
          05  CUST-NAME                  PIC X(30).
          05  CUST-ADDR                  PIC X(50).
```

Assume the record has been partially defined by the DEFINE RECORD command and additional fields need to be added:

```
alter table REC6 add columns after PART_JOB_GROUP
          3    PART-COUNT type INDEX-NAME unknown;

alter table REC6
        add columns before JOB-STATUS
        2   PART-ID unknown unique key
          3    PART-LOC type LOC-CODE
          3    PART-NAME type NAME-TYPE

        2   PART-JOB occurs 7 depending on PART-COUNT unknown
          3    PART-JOB-TIME type HOURS
          3    PART-JOB-PLACE type NAME-TYPE  ;
```

ALTER RELATION

Syntax:

```
alter relation [class] [{INTEGER1|many}] EC_NAME1 RC_NAME
          [{INTEGER2: INTEGER3|many} EC_NAME2
           [add  [migrates {KC_NAME [set {TAG_NAME1
           =TAG_NAME2}...]}]
           [keyword KEYWORD...]]
           [drop [migrates KC_NAME ]...
           [keyword KEYWORD...]];
```

Semantics:

1. To complete the above command the following elements must exist in Common Data Model:

   model
   link relation
   independent entity (EC_NAME1)
   dependent entity (EC_NAME2)

```
alter relation [class] [integer1|many] EC_NAME1 RC_NAME
     [integer2: (integer3|many)] EC_NAME2
```

1. The relation class is altered if any cardinality is altered.

2. Cardinality values default to the current cardinality for the link relation.

3. A warning message may be generated to indicate either left or right dependent cardinality too large. This will not halt processing but defaults the value to the current cardinality of the link relation. The right dependent cardinality (integer2) cannot be less than the left dependent cardinality (integer3).

4. Individual cardinalities may be changed without changing all the cardinalities.

5. The right dependent (integer3) cardinality cannot be zero.

```
      [add [migrates {KC_NAME [set {TAG_NAME1 =
TAG_NAME2}...]}
           [keyword KEYWORD...]]]
```

1. The following elements must exist in the Common Data Model:

   key for independent entity class
   key members for the independent entity
   attribute use class for each key member
   associated with the independent entity

2. The key cannot have been previously migrated to the dependent entity.

3. An attribute use class and an inherited attribute use class

for the independent entity is created for each key member of the independent entity migrated to the dependent entity.

4. If the set phrase is specified, TAG_NAME1 (the independent entity's tag name) is migrated with the new name of TAG_NAME1.

5. A complete relation occurrence is created.

6. The keyword is created in the Common Data Model if it doesn't already exist.

7. A relation keyword reference is created.

```
[drop [migrates KC_NAME]...
      [keyword KEYWORD...]];
```

1. The key migration will be dropped from each relation and entity class in the model.

2. The relation keyword reference is deleted.

3. The migrated link relation will not be dropped, when there exists a relation-to-set mapping.

Examples:

```
alter relation IND_EC_STORE RC_INVOICING
DEP_EC_CUSTOMER
      add migrates KC_ORDER set AC_INVOICE = AC_FORM
      drop keyword KC_ORDER;
```

ALTER UNION

Syntax:

```
alter union of record DB_NAME.REC_NAME

    [drop entity EC_NAME...]

    [add entity
           {EC_NAME when FIELD_NAME op 'STRING'
              {[and FIELD_NAME op 'STRING']...} ...   ;
```

Semantics:

1. This command shall add entities to a previously existing record union definition and drop entities from a record union.

2. The user is able to add an entity to an existing union with multiple discriminator conditions. To add or drop conditions from an existing entity of the union, the user should drop the entity and add the entity back with all the correct conditions.

3. The NDDL processor will not check for overlapping or nonsense conditions. The user is responsible.

4. For op in the union discriminator clause, the user should specify one of the following: =, >, <, <=, >=, != .

Examples:

```
alter union of record  DB2.RECX
     drop entity E3
     add entity E3 when DF1 <= 'C' ;

alter union of record DB2.RECY
     add entity E4 when DF5 = 'K' and DF6 = 'P'
                 E6 when DF5 = 'K' and DF6 = 'R' ;
```

an example of a nonsense condition:

```
alter union of record DB3.REC2
     add entity E1 when DF1 > 10 and DF1 < 10 ;
```

CHECK MODEL

Syntax:

check model MODEL_NAME;

Semantics:

1. MODEL_NAME must exist.

2. If the model can be checked to fulfill all rules, the model will be marked as checked.

3. The following rules are checked for the model:
   a. no non-specific link relations are allowed (independent cardinality greater than one).
   b. no incomplete link relations, (key has not been migrated).
   c. each entity must have at least one attribute use class.
   d. each owned attribute must have a domain and that domain must have a standard data type.
   e. a primary key must be defined for each entity.
   f. multiple keys of an entity must not be subsets of one another.
   g. no one to one relations.
   h. no dependency loops, e.g. A->B->C->D->B.
   i. at least one entity must exist in the model.
   j. alternate keys migrate through a link relation.
   k. category relation only has one category number.
   l. inherited keys are split among key and nonkey attributes in the dependent entity.

4. The following rules cannot be checked for the model:
   a. one to none or one relationships imply identical keys
   b. key uniqueness throughout the model is not checked, i.e. no two entities may have the same key unless they are related to each other with a one to none or one relation.

5. The output of this command will be generated to a user defined file or to the screen. Use "SET OUTPUT" to establish where the output is to be generated.

Examples:

```
check model A;
check model INTEGRATED_MODEL;
```

COMBINE ENTITY

Syntax:

```
combine entity EC_NAME_1 [from model MODEL_NAME]
        into EC_NAME_2
             [except [description] [alias] [keyword]];
```

Semantics:

1. EC_NAME2 cannot be a category entity of the generic entity EC_NAME1. (Combining category parent into any child is not allowed.)

2. If EC_NAME1 is a category entity, EC_NAME2 must be the immediate generic entity for EC_NAME1. (Combining category child into category parent is allowed.) If EC_NAME1 is the only category member then the category relation is dropped.

3. EC_NAME1 and EC_NAME2 cannot both be category members of the same category relation. (Combining category siblings is not allowed.)

4. If MODEL_NAME is not entered, an intra-model combine is assumed and EC_NAME_1 must not be the same as EC_NAME_2. The current model will be used.

5. The NDDL statements necessary to physically combine the two entities are generated on a file or the screen. Use "SET OUTPUT" to establish where the output is to be generated.

6. On an intra-model combine, any link relations between EC_NAME_1 and EC_NAME_2 will be dropped and EC_NAME_1 will be dropped.

7. All owned attributes of EC_NAME_1 and their aliases, keywords and descriptions are generated for EC_NAME_2 unless they already exist for EC_NAME_2, or the corresponding keyword appears in the EXCEPT clause.

8. All entity names, keywords and descriptions of EC_NAME_1 are generated as aliases, keywords and descriptions of EC_NAME_2 unless they already exist for EC_NAME_2, or the corresponding keyword appears in the EXCEPT clause.

9. All relations in which EC_NAME_1 is the dependent entity or category entity are generated for EC_NAME_2, provided the independent entities or generic entities in the relations already exist in the current model and the relation names are not already associated with EC_NAME_2. The keys of the independent entities are migrated to EC_NAME_2 and all keys of EC_NAME_1 are generated for EC_NAME_2.

10. All relations in which EC_NAME_1 is the independent entity or generic entity are generated for EC_NAME_2, provided the dependent entities or category entities already exist in the current model and the relation names are not already associated with EC_NAME_2. The key of EC_NAME_1 is migrated to the dependent entities but is not added as key to the dependent entities.

11. The 'EXCEPT' clause may be omitted or entered for any type of copy. If the except clause is entered, at least one keyword must be specified. Any or all of the three keywords may be used. If used, the keyword must appear in the order indicated. If 'description' is entered, no descriptions for entities, attributes, and relations will be generated or copied. If 'alias' is entered, no aliases for entities and attributes will be generated or copied. If 'keyword' is specified, no keywords for entities, attributes and relations will be generated or copied.

Generated Commands

The generated NDDL commands should be examined for potential run-time errors.

1. If the entity being combined has inherited attributes then the generated NDDL must be changed either to add the inherited attributes to the new entity as owned attributes, or all references to the inherited attributes must be deleted
from KEY CLASS and MIGRATES clauses.

2. A create/alter entity command may attempt to add an owned attribute when the attribute is already owned by another entity in the target model. The modeler must decide which entity should own the attribute and change the NDDL accordingly.

NOTE: When an attribute is added to an entity and the attribute already exists in the target model, a comment is generated in the NDDL command following the attribute. The comment is:

```
/* ATTRIBUTE MAY BE OWNED IN TARGET MODEL */
```

The comment will not cause an NDDL syntax error.

Examples:

```
combine entity ENT8 into ENT_B except description;

combine entity ENT_B from model SPARKY into ENT_B;
```

COMMIT

Syntax:

```
commit;
```

Semantics:

1. This command will commit all changes made to the CDM since the last commit point.

2. If the current setting of commit mode is "automatic", this command will have no effect since any changes would have been committed upon the completion of the previous command or, if in error, rolled back.

Syntax:

```
compare model MODEL_NAME_1 with MODEL_NAME_2
    [except alias];
```

Semantics:

1. MODEL_NAME_1 and MODEL_NAME_2 must both exist.

2. Similarities or points of correspondence of the two models will be reported:

   a. entity names correspond, (match identically) either through primary name or alias
   b. attribute names correspond, either through primary name or alias
   c. link relation names correspond either through primary name or alias
   d. entity keywords correspond
   e. attribute keywords correspond
   f. relation keywords correspond
   g. category relation names, discriminating attribute names, number of category entities correspond only when both relations are complete or both incomplete.

3. If "except aliases" is specified, similarities will be reported for entity, attribute and relation class names when they correspond (match identically) on primary name.

4. The output will be generated to a user specified file or the screen. Use "SET OUTPUT" to establish the output desired.

Examples:

```
compare model A with model B;
```

COPY ATTRIBUTE

Syntax:

```
copy attribute ATTR_NAME_1 [from model MODEL_NAME]
       [to ATTR_NAME_2] [directly]
       [except [description] [alias] [keyword]];
```

Semantics:

1. To complete the command the following elements must exist in the Common Data Model:

   attribute being copied
   current model

2. If "DIRECTLY" was not specified, the user must have already done the 'SET OUTPUT' command. 'SET OUTPUT' will cause the copy to go to the screen or to a file. (See SET OUTPUT). If "DIRECTLY" was specified, the copy command will physically add the new attribute to the current model, as opposed to generating NDDL commands.

3. If the "FROM" clause is omitted, an intra-model copy is assumed and the following rules apply: when the copy is "DIRECTLY", the CDM will be updated; otherwise, NDDL commands will be generated and outputted to a specified file or screen.

   3.1 ATTR_NAME_2 must be entered and may not be the same as ATTR_NAME_1.

   3.2 If the copy is DIRECTLY, the following applies:

      3.2.1 A new attribute class is built for the current model.

      3.2.2 All of ATTR_NAME_1's descriptions and keywords are created for ATTR_NAME_2, unless excepted.

      3.2.3 Attribute alias names are not copied because an alias name uniquely identifies only an attribute in the model.

   3.3 If the copy is not DIRECTLY; i.e., NDDL commands are to be generated, the following applies:

      3.3.1 NDDL to create the attribute class is generated.

      3.3.2 Keywords, aliases and descriptions of ATTR_NAME_1 are generated for ATTR_NAME_2, unless excepted.

      3.3.3 The NDDL generated will not execute successfully, if aliases have not been excepted.

4. If the 'FROM' clause is entered, the following rules apply:

   4.1 An inter-model copy is assumed.

   4.2 The attribute class being copied must not exist in the current model.

   4.3 If ATTR_NAME_2 is omitted, it defaults to ATTR_NAME_1.

   4.4 If the copy is DIRECTLY, the following applies:

      4.4.1 A new attribute class is built for the current model.

      4.4.2 Keywords, aliases and descriptions of ATTR_NAME_1 are copied for ATTR_NAME_2, provided they do not previously exist and have not been excepted.

   4.5 If the copy is not DIRECTLY, the following applies:

      4.5.1 NDDL is generated to create an attribute class for the current model.

      4.5.2 Also, NDDL commands to copy the attributes, descriptions, aliases and keywords are generated, unless excepted.

Examples:

```
copy attribute A1 from Model_1 to A4;
copy attribute A3 from Model_2 directly;
copy attribute A5 to A6
except description alias;
```

COPY DATABASE

Syntax:

```
copy {database} {DB_NAME...} [except description];
     {pcb     } {all       }
```

Semantics:

1. NDDL commands will be generated to copy a specified database or all databases in the CDM.
2. The commands will be generated, ordered by database name, and then by records in the database.
3. This command will not change the current database.
4. For an ORACLE database, the password will be copied.
5. An ALTER DATABASE command will be generated after each database's definition has been copied.
6. For a VAX-11/IDMS/IDS-II database, the schema, subschema, areas, and directory name where the VAX-11 database is located will be copied.
7. For an IMS pcb, the psb name, start position and feedback length will be copied.
8. The specification of null values will be copied.
9. The two character field representing the NTM directory of generated RP's will be copied.
10. All the records, data fields, and sets belonging in the database are copied.
11. Description text for the database, records, data fields, and sets will be copied if the keywords "except description" are not specified.
12. The generated NDDL commands will be appended to the previously named file or outputted to the screen.

Examples:

```
copy database all except description;
```

Generated NDDL:

```
DEFINE VAX-11 DATABASE NAMED COD_DB ON HOST IBM
  WITH SCHEMA S1 AND SUBSCHEMA SS1
  AREAS A1 A2
  STORES CHARACTER NULL AS ZEROS
```

```
        INTEGER NULL AS ZEROS
  NTM DIRECTORY GR;

ALTER DATABASE COD_DB;
DEFINE RECORD REC1
  IN AREAS A1 WITH FIELDS
  F1 TYPE NAME_TYPE
  F2 TYPE ADDR_TYPE
  F3 TYPE EMP_DATA key;

DEFINE RECORD REC2
  IN AREAS A2 WITH FIELDS
  1  PART_JOB_GROUP          UNKNOWN
    2  JOB_STATUS            TYPE_CODE_DATE_TYPE
    2  JOB_INFO              REDEFINES JOB_STATUS
      3  JOB_ID              TYPE CODE_TYPE
      3  JOB_DATE            TYPE DATE_TYPE
  1  PART_CUST_GROUP         OCCURS 100 INDEXED BY
                          X-X UNKNOWN
    2  CUST_NAME             TYPE NAME_TYPE
    2  CUST_ADDR             TYPE ADDR_TYPE;

DEFINE SET SETA
  RELATING REC1 TO REC2 REQUIRED;


DEFINE IMS PCB NAMED IMS_1 ON HOST IBM
  WITH POSITION 1 IN PSB SS_PSB
  FEEDBACK LENGTH 20
  STORES CHARACTER NULL as SPACES
        INTEGER NULL AS LOW_VALUES
  NTM DIRECTORY GR ;

ALTER DATABASE IMS1;
DEFINE RECORD S1
  X2  TYPE  CODE_TYPE
  FILLER  5
  FILLER 10
  X5  TYPE  JOB_TYPE  UNKNOWN ;

DEFINE RECORD S2
  A1  TYPE  NAME_TYPE
  A2  TYPE  JOB_TYPE  ;

DEFINE PATH
  FROM S1 TO S2 REQUIRED;
```

COPY DBMS

Syntax:

```
copy dbms ( DBMS_NAME..) [[include database] [except
                                              description]];
                  ( all        )
```

Semantics:

1. NDDL commands will be generated to copy a specified DBMS or all DBMS in the CDM.

2. Any hosts that are associated with the DBMS will also be copied.

3. The generated commands will be appended to the previously named file or outputted to the screen.

4. All databases belonging to the DBMS will be copied, if the keywords "include database" are specified.

5. All description text for the database, records, sets and fields will be copied, if the keywords "except description" are not specified.

Examples:

```
copy dbms all;
```

Generated NDDL:

```
DEFINE DBMS XX MODEL R ON HOST VAX IBM;
DEFINE DBMS YY MODEL N;
DEFINE DBMS ZZ MODEL I ON HOST NCR;
```

COPY DESCRIPTION

Syntax:

```
copy description {DESC_TYPE...} of object_type
                 { all         }

OBJECT_NAME [to OBJECT_NAME] [from model FROM_MODEL]
            [directly] ;
```

Semantics:

1. Following is a list of objects for which description text may be copied and must exist in the Common Data Model:

OBJECT TYPE	OBJECT NAME(S)
attribute	AC_NAME
entity	EC_NAME
relation	EC_NAME1 RC_NAME [EC_NAME2]
database	DB_NAME
record	DB_NAME REC_NAME
datafield	DB_NAME REC_NAME FIELD_NAME
set	DB_NAME SET_NAME
dataitem	VIEW_NAME DI_NAME
view	VIEW_NAME
userdatatype	DOMAIN_NAME DATA_TYPE_NAME
domain	DOMAIN_NAME
keyword	KWORD_NAME
model	MODEL_NAME
host	HOST_NAME

2. If attribute, entity, or relation descriptions are going to be copied, a current model must be established for the session.

```
copy description {DESC_TYPE...}
                 { all         }
```

1. The user has the option to specify which description type(s) he wishes to copy, or the user can copy all description types by using the all option.

```
of object_type OBJECT_NAME [to OBJECT_NAME]
```

1. The object_type must be one of the object types listed above.

2. The user must enter an object_name or a combination of object names that correspond to the object_type selected.

3. If object type is relation, EC_NAME2 RC_NAME EC_NAME2 signifies a link relation and EC_NAME1 RC_NAME signifies a category relation.

4. The to object_name clause is optional. When the clause is specified, the name of the object being copied is changed to the to object_name, either directly or on the generated DESCRIBE command.

[from model FROM_MODEL]

1. The from_model may be specified if attribute, entity, or relation descriptions are being copied inter-model. Otherwise, the current model is used.

[directly]

1. If directly is specified, the description will be copied directly to the CDM.

2. If directly is not specified, the user must have already done a SET OUTPUT command. The generated NDDL commands will be written to the file specified in the SET OUTPUT command. If the object_type is an entity, relation, or attribute, an ALTER MODEL command is also generated.

Example:

```
copy description DEFINITION of attribute AC_NO from model
INTEGRATED_MODEL;
```

Generated NDDL:

```
ALTER MODEL INTEGRATED_MODEL;
DESCRIBE DEFINITION OF ATTRIBUTE AC_NO
"This is the definition of attribute AC_NO";
```

Other Examples:

```
copy description all of host VAX to CYBER directly;
copy description DEFINITION SOURCE of entity ENTITY_CLASS to
  entity ENTITY_CL from
         model INTEGRATED_MODEL directly;
copy description SOURCE of view USER_VIEW to VIEW_OF_USERS;
copy description DEFINITION of relation ENTITY_CLASS HAS
  ENTITY_NAME to ENTITY_LASS MAKES ENTITY_GLASS
         directly;
copy description all of database ORACLE;
```

COPY DESCRIPTION TYPE

Syntax:

```
copy description type (DESC_TYPE ...);
                      (all          )
```

Semantics:

1. NDDL commands will be generated to create a specified description type or all description types stored in the CDM.

2. The generated NDDL commands will be appended to the previously named file or outputted to the screen as established by a previous SET OUTPUT command.

Examples:

```
copy description type DEFINITION SOURCE RULE;
```

Generated NDDL:

```
CREATE DESCRIPTION TYPE DEFINITION;
CREATE DESCRIPTION TYPE SOURCE;
CREATE DESCRIPTION TYPE RULE;
```

Syntax:

```
copy domain      (DOM_NAME...) [except [description]
[userdatatype]]; (all          )
```

Semantics:

1. NDDL commands will be generated to copy a specified domain or all domains in the CDM.

2. The standard and user defined data types will be copied if the clause "except userdatatype" is not specified.

3. All textual description for the domain and data types are also copied, if the clause "except description" is not specified.

4. The domain values and ranges are copied.

5. The generated NDDL commands will be appended to the previously named file or outputted to the screen.

Examples:

```
copy domain   EMP_ADDR  except description;
```

Generated NDDL:

```
CREATE DOMAIN EMP_ADDR STANDARD
  TYPE  STRING_60        CHARACTER  60
  TYPE  ZIP_CODE         INTEGER     5
  RANGE '1' thru '99999' ;
```

COPY ENTITY

Syntax:

```
copy entity [class] EC_NAME_1 [from model MODEL_NAME]
       [to EC_NAME_2]   [directly]
       [with  {structure} ]
              {relation}
       [except [description] [alias] [keyword] [key]
         [nonkey]
       [attribute] [DEP_ENT_NAME...]] ;
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   entity being copied
   current model

2. If 'DIRECTLY' was not specified, the user must have already done a 'SET OUTPUT' command. 'SET OUTPUT' will cause the copy to go to the screen or to a file (SEE SET OUTPUT). If 'DIRECTLY' was specified, the copy command will update the CDM as opposed to generating NDDL commands on file or screen.

3. If the 'FROM' clause is omitted, an intra-model copy is assumed and the following rules apply.

   3.1 EC_NAME_2 must be entered and may not be the same as EC_NAME_1.

   3.2 The 'WITH' clause must be omitted.

   3.3 If the copy is DIRECTLY (Intra-model, Directly), the following applies:

   3.3.1 The allowed exceptions are keywords and descriptions.

   3.3.2 A new entity is built for the current model.

   3.3.3 No attributes are built, as these attributes already exist in the current model.

   3.3.4 Entity alias names will not be built because an alias uniquely identifies only one entity in the model.

   3.4 If the copy is not DIRECTLY; i.e., NDDL commands are to be generated (Intra-model, On file), the following applies:

   3.4.1 The allowed exceptions are descriptions, aliases, keywords and attributes.

3.4.2 NDDL to create an entity is generated, along with its owned attributes.

3.4.3 NDDL to associate keywords, aliases and descriptions of the entity and attributes are generated.

3.4.4 The NDDL generated will not execute successfully, unless aliases and attributes have been excepted.

3.5 Keys associated with the entity are not copied in an intra-model copy.

4. If the 'FROM' clause is entered, an inter-model copy is assumed and the following rules apply:

4.1 The entity being copied must not exist in the current model.

4.2 If EC_NAME_2 is omitted, the name of the new entity will be the same as EC_NAME_1.

4.3 If the copy is DIRECTLY (Inter-model, Directly), the following applies:

4.3.1 The 'WITH' clause is prohibited.

4.3.2 The allowed exceptions are descriptions, aliases, keywords, keys, nonkeys, and attributes.

4.3.3 A new entity is built for the current model along with its keywords, descriptions, and aliases unless excepted.

4.3.4 New attributes are built for the current model for each of the owned attributes of the copied entity unless attributes are excepted. All keywords, descriptions, and aliases are also built for each attribute unless excepted.

4.3.5 All keys are also built for the current model for each key of the entity unless excepted. Key members that are owned and key in the FROM entity are copied to the TO entity. If attributes are excepted, nonkeys and keys will automatically be excepted.

4.4 If the copy is not DIRECTLY and the 'WITH' clause was not specified (Inter-model, On file, Simple), the following applies:

4.4.1 NDDL will be generated to create a new entity in the current model.

4.4.2 The allowed exceptions are descriptions, aliases, keywords, keys, nonkeys, and attributes.

4.4.3 All keywords, aliases, and descriptions associated with the entity being copied are generated unless excepted.

4.4.4 All attributes owned by the entity being copied are generated unless attributes are excepted. Also, all the descriptions, aliases, and keywords for each attribute are generated unless excepted. If attributes are excepted, keys and nonkeys default to excepted, too.

4.4.5 All keys and key members belonging to the entity being copied are generated unless excepted. If keys are excepted, the copied attributes are generated as owned and not key.

4.4.6 If nonkeys are excepted, only attributes that are members of a key are copied.

4.5 If the WITH option is chosen; i.e., with Structure or with Relation

4.5.1 All possible exceptions are allowed in this case.

4.5.2 If attributes are excepted, the create attribute command is not generated, nor the alter entity owned attribute clause. In addition, the key clause is not generated in the alter entity command, nor the migrates..set clause of the create relation command.

4.5.3 If nonkeys are excepted, the NDDL commands to alter entity..add owned attribute clause are not generated for attributes that are not key members.

4.5.4 All keys which were migrated in the relations being copied are migrated unless keys are excepted. The same role names used in the set phrase of the relation being copied are generated for the current model.

4.5.5 Keys will not be migrated in the current model for relations using entities that were specified as part of the excepted entity list.

4.6 If the WITH RELATION option is chosen (Inter-model, On file, with relation), the following applies:

4.6.1 All the relations in which the entity being copied is the dependent entity or category entity are generated for the current model, provided the independent entities or generic entities in the relations exist in the current model.

4.6.2 All the relations in which the entity being copied is the independent entity or generic entity , are generated for the current model, provided the dependent entities or category entities in the relations exist in the current

model and the dependent entity or category entity was not specified as part of the excepted entity list.

4.6.3 If it is a complete relation, NDDL to migrate the key is generated.

4.6.4 If the link relation has not migrated a key, then only the relation is copied.

4.7 If the WITH STRUCTURE option is chosen (Inter-model, On File, with structure), the following applies:

4.7.1 The tree structure dependent on the entity being copied is generated for the current model. This includes the entities; their keywords, aliases, descriptions, owned attributes, keys, and key members unless excepted; both link and category relations, their migrated keys and set names unless excepted.

4.7.2 If an excepted entity list is specified, nothing is generated for each excepted entity or any entity that is part of its dependent tree structure.

Generated Commands

The generated NDDL commands should be examined for potential run-time errors.

1. If the entity being copied has inherited attributes then the generated NDDL must be changed either to add the inherited attributes to the new entity as owned attributes, or all references to the inherited attributes must be deleted from 'KEY CLASS' and 'MIGRATES' clauses.

2. If a tree structure is being copied, then the create commands for any entities, attributes, aliases and relations which already exist in the current model must be deleted.

3. A create/alter entity command may attempt to add an owned attribute to an entity when the attribute is already owned by another entity in the target model. The modeler must decide which entity should own the attribute and change the NDDL accordingly.

4. A CREATE/ALTER category commands will be generated but may need to be changed or removed if attribute, keys and nonkeys are excepted.

NOTES: 1. When an attribute is added to an entity and the attribute already exists in the target model, a comment is generated in the NDDL command preceding the attribute. The comment is:

/* ATTRIBUTE IS CURRENTLY OWNED IN THE TARGET MODEL*/

2. When an alias is specified for an attribute and the alias already exists in the target model, a comment is generated in the NDDL command preceding the alias. The comment is:

```
/* NDDL WON'T EXECUTE:  ATTRIBUTE ALIAS EXISTS IN
TARGET MODEL */
```

3. When an alias is specified for an entity and the alias already exists in the target model, a comment is generated in the NDDL command preceding the alias. The comment is:

```
/* NDDL WON'T EXECUTE:  ENTITY ALIAS EXISTS IN TARGET
MODEL */
```

4. When a dependent entity is added to the target model and the entity already exists in the target model, a comment is generated in the NDDL command preceding the entity. The comment is:

```
/* NDDL WON'T EXECUTE:  ENTITY IS IN TARGET MODEL */
```

5. When generating the "key clause", if a key class member is inherited in an Inter-model, On File, Simple copy, a comment is generated following the Alter Entity...Add key command. The comment is:

```
/* NDDL WILL NOT EXECUTE:  KEY CLASS MEMBER IS
INHERITED */
```

6. If an Intra-MODEL, On File copy is done, a comment is generated on the top of the file which states:

```
/* THIS FILE WILL EXECUTE ONLY IF ATTRIBUTES AND
ALIASES ARE EXCEPTED */
```

7. These comments will not cause NDDL syntax errors.

Examples:

```
copy entity INVOICE to NEWINVOICE except description alias
        keyword;
copy entity INVOICE from model ABC_COMPANY
        with structure except alias;
copy entity INVOICE to from model ABC_COMPANY directly
        except attribute key;
```

COPY HOST

Syntax:

```
copy host {HOST_NAME...} [except [description] [psb]];
     {all          }
```

Semantics:

1. NDDL commands will be generated to copy a specified host or all hosts in the CDM.

2. All description text for the host will be copied, if the keywords "except description" are not specified.

3. Any PSB's associated with the host will be copied, if the keywords "except psb" are not specified.

4. The generated NDDL commands will be appended to the previously named file or outputted to the screen.

Examples:

```
copy host IBM except description;
```

Generated NDDL:

```
DEFINE HOST IBM ;
DEFINE PSB SS_PSB ON HOST IBM;.
DEFINE PSB ABC ON HOST IBM;
```

Syntax:

```
copy map
     [for entity     {EC_NAME...}
                     {all       }
     [except [partition] [union]          ]

     [for relation   {EC_NAME1 RC_NAME EC_NAME2...}       ]
                     {all                         }
     [for category   {EC_NAME1 RC_NAME...}
                     {all              }
     [for record     {DB_NAME.REC_NAME...}
                     {all                 }
     [except [partition] [union]          ]

     [for set        {DB_NAME.SET_NAME...}                ]     ;
                     {all                 }
```

Semantics:

1. NDDL commands will be generated to copy mapping definitions between the two schemas, conceptual and internal. The map commands can be copied for conceptual objects (through category entity or relation) or copied for internal objects (through record or set).

2. If EC_NAME, EC_NAME1 or EC_NAME2 is entered as the alias name, the copy map will generate the map commands using the primary entity name.

3. [Copy Map for entity {EC_NAME} [except [partition] [union]]
   {all   }

   3.1 All mappings for a specified entity or mapping definitions for all entities in the CDM will be copied.

   3.2 If the option "all" is chosen, the entity is ordered by entity name.

   3.3 If the entity is horizontally partitioned among two or more record types, the NDDL command to create the partition is also copied, unless the clause "except partition" is specified.

   3.4 If the entity is part of a record union, the NDDL command to create the union is also generated, unless the clause "except union" is specified.

   3.5 If any tag in the entity is an input or output parameter of an algorithm, the NDDL command to define the complex mapping algorithm is generated.

   3.6 If any tag in the entity is mapped to sets, the AUC to SET create map commands will be generated.

   3.7 Warning messages are issued for entities that are not mapped.

Example:

```
copy map for entity PART ;
```

Generated NDDL:

```
CREATE PARTITION 1 OF ENTITY PART
    TO RECORDS
       DB1.REC1
       DB2.REC2;
CREATE   MAP    PART.PART_SIZE    FOR  PREFERENCE 1
    TO FIELD  DB1.REC1.DF1 ;
CREATE MAP      PART.PART_SIZE    FOR  PREFERENCE 1
    TO FIELD  DB1.REC2.DF1 ;
CREATE MAP      PART.PART_LOCK     FOR  PREFERENCE 1
    TO FIELD  DB1.REC1.DF2 ;
CREATE MAP      PART.PART_ORDER   FOR  PREFERENCE 1
    TO FIELD  DB1.REC1.DF3 ;
DEFINE ALGORITHM MOD23 FOR RETRIEVAL
    USING PARAMETERS
P1 FROM    DATAFIELD     PARTDB.CUST_ORDER.DF1
P2 FROM    DATAFIELD     PARTDB.CUST_ORDER.DF2
P3 TO      ATTRIBUTE     PART.STATUS
    STATUS ;
```

4. [copy map for relation {EC_NAME1 RC_NAME EC_NAME2...}]
                          {all   }

4.1 The map between link relation and a record set is copied for a specified link relation of all link relation.

4.2 If the "all" option is chosen, the link relation to set maps will be ordered by independent entity, relation name and dependent entity.

Example:

```
copy map for relation all;
```

Generated NDDL:

```
CREATE MAP EMPLOYEE POSSESSES SKILLS
    TO SET SKILLDB.SETA.RECA
    TO SET SKILLDB.SETB ;
CREATE MAP KEY_CLASS HAS_MANY KEY_CLASS_MEMBERS
    TO SET TOTDB.CLASSES ;
```

5. Copy map [for category {EC_NAME1 RC_NAME...}]
                          {all   }

5.1 The map between category relation and a record set is copied for a specified category relation of all category relations.

5.2 If the "all" option is chosen, the category relation to set maps will be ordered by the generic entity and relation name.

6. Copy map [for record {DB_NAME.REC_NAME...}
{all }
[except [union] [partition]] ;

6.1 A map is copied for either a specified record or all in the CDM. Mapping definitions exist between
1) data fields of the record and a tag, and
2) a set, where the record is the owner, and a tag.

6.2 If the record represents a union between two or more entities, the record union is also copied, unless the "except union" clause is specified.

6.3 If the "all" option is chosen, the record maps are ordered by record name.

6.4 If record is part of a union, the NDDL command to create the union is generated, unless "except union" is specified.

6.5 If record is part of a partition, the NDDL command to create the partition is generated, unless "except partition" is specified.

6.6 Warning messages are issued for records that are not mapped.

Example:

```
Copy map for record CDM.COMPLEX_MAPPING_PARM ;
```

Generated NDDL:

```
CREATE UNION OF RECORD CDM.COMPLEX_MAPPING_PARM
   TO ENTITY
       AUC_PARM     WHEN UNION_DISC  =  "TAG"
       RT_PARM      WHEN UNION_DISC  =  "RT" ;

CREATE MAP     AUC_PARM.MOD_ID    FOR    PREFERENCE 1
   TO FIELD CDM.COMPLEX_MAPPING_PARM.MOD_ID ;
CREATE MAP     RT_PARM.MOD_ID     FOR    PREFERENCE 1
   TO FIELD CDM.COMPLEX_MAPPING_PARM.MOD_ID ;
CREATE MAP     AUC_PARM.PARM_ID   FOR    PREFERENCE 1
   TO FIELD CDM.COMPLEX_MAPPING_PARM.PARM_ID ;
CREATE MAP     RT_PART.PARM_ID    FOR    PREFERENCE 1
   TO FIELD CDM.COMPLEX_MAPPING_PARM.PARM_ID ;
CREATE MAP     AUC-PARM.TAG_NO    FOR    PREFERENCE 1
   TO FIELD CDM.COMPLEX_MAPPING_PARM.TAG_NO ;
CREATE MAP      RT_PARM.RT_NO     FOR    PREFERENCE 1
   TO FIELD CDM.COMPLEX_MAPPING_PARM.RT_NO ;
```

7. [Copy map for set {DB_NAME.SET_NAME...}];
{all }

7.1 The mapping between a set and link or category relation is copied for a specified set or all the sets in the CDM.

7.2 If the "all" option is chosen, the set maps copied will be ordered by set name.

7.3 AUC to SET mappings cannot be generated by copy map for set since the mappings involve additional sets other than the one the user specified. User would need to use copy map for entity of the appropriate attribute.

7.4 Warning messages are issued for sets that are not mapped.

Example:

```
Copy map for set DB1.SETA ;
```

Generated NDDL:

```
CREATE MAP EMP POSSESSES SKILL
    TO SET DB1.SETA ;
```

Syntax:

```
copy model [from model MODEL_NAME] to NEW_MODEL
    [except [description] [alias] [keyword] [attribute]
    [key] [nonkey] [entity] [relation] [inherited]
    [ENT_NAME....]];
```

Semantics:

1. To complete the above command the following element must exist in the Common Data Model:

   model being copied

2. The user must have already done a 'SET OUTPUT' command. 'SET OUTPUT' will cause the copy to go to the screen or to a file (SEE SET OUTPUT).

3. If the 'FROM' clause is entered, MODEL_NAME must exist. If it is omitted, the current model will be copied.

4. NEW_MODEL must not exist.

5. The model being copied may not contain any dependency loops. Use 'CHECK_MODEL' to determine if a dependency loop is present.

6. NDDL commands are generated in the proper sequence to create a new model containing all the entities, owned attributes, inherited attributes, keys, key members, link relations, category relations, complete relations, aliases, keywords and descriptions in the model being copied unless any of the above are excepted.

7. Aliases, keywords, descriptions, entities, nonkeys, keys, inherited attributes, relations, attributes and/or a list of specific entities will be excluded from the generated NDDL if the corresponding keyword appears in the 'EXCEPT' clause.

8. If descriptions are excepted, no 'DESCRIBE' clauses will be generated for the attributes, entities or relations of the model.

9. If aliases are excepted, no 'CREATE ALIAS' clauses will be generated for the attributes or entities of the model.

10. If keywords are excepted, no 'KEYWORD' clauses will be generated for the attributes, entities or relations of the model.

11. If attributes are excepted:

    a. No 'CREATE ATTRIBUTE' commands are generated.
    b. The 'CREATE ENTITY' command will be generated without the 'OWNED ATTRIBUTE' or the 'ADD KEY' clause.
    c. The 'CREATE RELATION' commands, if any, will not have a

'MIGRATES...SET...' clause.
d. The 'CREATE CATEGORY' commands, if any, will be generated but needs to be checked for the existence of the discriminating attribute.

12. If keys are excepted:

a. 'CREATE ATTRIBUTE' commands will be generated for the keyed attributes, if attributes are not excepted.
b. The 'CREATE ENTITY' command will have the 'OWNED' clause, if any, but no 'KEY' clause.
c. The 'CREATE RELATION' command, if any, will not have a 'MIGRATES...SET...' clause.
d. The 'CREATE CATEGORY' command, if any, will be generated but needs to be checked for the existence of primary keys.

13. If nonkeys are excepted:

a. No 'CREATE ATTRIBUTE' clause will be generated if the attribute is not a key member.
b. No 'OWNED ATTRIBUTE' clause will be generated if the attribute is not a key member.
c. The 'CREATE CATEGORY' command, if any, will be generated but needs to be checked for the existence of the discriminating attribute.

14. If entities are excepted, a comprehensive list of attributes of the model will be generated.

15. If relations are excepted:

a. The model will result in a list of entities, attributes, keys and key class members.
b. No 'CREATE RELATION' commands will be generated, nor will the 'MIGRATES...SET...' clause.
c. No 'CREATE CATEGORY' commands will be generated.

16. If inherited attributes are excepted:

a. The 'CREATE RELATION' command, if any, will not have a 'MIGRATES...SET...' clause; i.e., NDDL to migrate the key class will not be generated.
b. The 'CREATE CATEGORY' command, if any, will be generated but needs to be checked for the existence of primary keys.

17. If a list of specific entities (ENT_NAME) are excepted:

a. The command will behave as if that dependent or category entity did not exist in the FROM model.
b. The link relations between the subject entity and the listed dependent entity are not copied.
c. The category relations between the subject entity and the listed category entity are not copied.
d. The excepted entity and any entity that is part of its dependent tree structure along with the associated attributes, keys, key members, aliases, and descriptions will not be copied.

18. If the combination of key, nonkey and entity are excepted, a comprehensive list of all attributes that are not owned by any entity in the model will be generated.

Examples:

```
copy model from model M1 to M2;

copy model to M2 except relations;

copy model to M2 except description alias keyword E1  E2;
```

COPY MODULE

Syntax:

```
copy module  {MOD_NAME...}  ;
             {all          }
```

Semantics:

1. NDDL commands will be generated to copy a specified module or all modules stored in the CDM through the Define Module..command, as opposed to those tracked by the precompiler.

2. The modules parameters, language the module is written in, and user defined data types describing the parameters are copied.

3. The generated NDDL command will be appended to the previously named file or outputted to the screen.

Examples:

```
copy module  MOD1 ;
```

Generated NDDL:

```
DEFINE MODULE MOD1 IN COBOL
  PARAMETERS
    IN_REC         TYPE   PART_RECORD
    PART_WEIGHT    TYPE   KILOS
    STD_STATUS     TYPE   RET_STATUS ;
```

COPY RECORD

Syntax:

```
copy record {REC_NAME...}[of database DB_NAME]
            {all        }
     [except description] ;
```

Semantics:

1. NDDL commands will be generated to copy a specified record or all records belonging to a specified database.

2. If the "all" option is chosen, the records of the current database are copied.

3. The "of database" clause may be omitted if a current database has been established for the session.

4. The record layout is copied in correct sequence, listing the level of group and subcomponent data fields. All the information pertaining to each data field is copied, whether it is repeating, redefined, indexed or key.

5. If the "all" option is chosen, the records copied will be ordered by record name.

6. All textual description for the record and its data fields are copied if the keywords "except description" are not specified.

7. The generated NDDL commands will be appended to the previously named file or outputted to the screen.

Examples:

```
copy record REC1 of database TOT_DB except description;
```

Generated NDDL:

```
ALTER DATABASE TOT_DB;
DEFINE RECORD REC1 WITH ELEMENTS
   1 DATA                    TYPE   X100
   1 RECA                    REDEFINES  DATA
    2 AAA1                   TYPE   X50
    2 AAA2                   TYPE   X20
    2 FILLER 30
   1 RECB                    REDEFINES  DATA
    2 BBB1                   TYPE   X100
      3  BIAA                OCCURS 10 INDEXED BY X_INDEX
                                 UNKNOWN
       4  BIAB                TYPE   X10;
```

COPY SET

Syntax:

```
copy set          {SET_NAME...}  [of database DB_NAME]
                  {all         }
          [except description];
```

Semantics:

1. NDDL commands will be generated to copy a specified set or all sets belonging to a specified database.

2. The "of database DB_NAME" clause may be omitted if a current database has been established for the session.

3. If the "all" option is chosen, all the sets of the current database are copied.

4. If the "all" option is chosen, the sets copied are ordered by set name.

5. The textual description for the set will be copied if the keywords "except description" are not specified.

6. The generated NDDL commands will be appended to the previously named file or outputted to the screen.

Examples:

```
ALTER DATABASE  TOT_DB ;
Copy set CONTROLLED_BY  ;
```

Generated NDDL

```
          ALTER DATABASE  TOT_DB  ;
          DEFINE SET CONTROLLED_BY ;
            RELATING SHOP_SUPPLY to SHOP_REQUESTS
            LINKED BY SHOP_NO ;
          DESCRIBE SOURCE OF SET CONTROLLED_BY
            "CONTROLLED BY IS THE RELATION BETWEEN MASTER
RECORD
            SHOP SUPPLY AND VARIABLE RECORD SHOP REQUESTS" ;
```

COPY VIEW

Syntax:

```
copy view    {VIEW_NAME...}
             {all         }
[except [description] [algorithm]];
```

Semantics:

1. NDDL commands will be generated to copy a specified view or all the views known to the CDM.

2. All textual description for the view will be copied if the clause "except description" is not specified.

3. If the data item is an input or output parameter of an algorithm, the algorithm will be copied, unless the clause "except algorithm" is specified.

4. The generated NDDL commands will be appended to the previously named file or outputted to the screen.

Examples:

```
copy view  SUPPLIES  except description ;
```

Generated NDDL:

```
CREATE VIEW SUPPLIES
               DATA ITEM   SUPPLIER   SUPPLIER_TYPE
               DATA ITEM   LOCATION   LOCATION_TYPE
               DATA ITEM   REQUESTOR  REQUESTOR_TYPE

           AS SELECT   PURCHASE_ORDER.ITEM_SUPPLIER
                       DELIVERY.DELIVERY_LOCATION
                       MATERIAL_REQUEST.REQUESTING_SEC

              FROM     PURCHASE_ORDER
                       DELIVERY
                       MATERIAL_REQUEST

              WHERE    PURCHASE_ORDER   SPECIFIED  DELIVERY AND
                       MATERIAL_REQUEST REQUESTS
   PURCHASE_ORDER ;

           DEFINE ALGORITHM MOD2 FOR UPDATE
              USING PARAMETERS
              IN_REC   FROM   DATAITEM   SUPPLIES.SUPPLIER
              OUT_REC  TO     ATTRIBUTE
   PURCHASE_ORDER.ITEM_SUPPLIER
              STATUS ;
```

CREATE ALIAS

Syntax:

```
create alias ( entity EC_NAME1 [is] EC_NAME2
             | attribute AC_NAME1 [is] AC_NAME2 ) ;
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:
   entity class or
   attribute class

2. EC_NAME2 and AC_NAME2 are the names to be created as aliases for entity and attribute respectively.

Examples:

```
create alias entity CUSTOMER is CUST;
create alias attribute ORDER_NUMBER is ORDNO;
```

Syntax:

```
create attribute [class] AC_NAME [domain DOMAIN_NAME]
          [keyword KEYWORD ...];
```

Semantics:

```
create attribute [class] AC_NAME
```

1. To complete the above command the following element must exist in the Common Data Model: the current model.

2. A new attribute class is created for the current model.

[domain DOMAIN_NAME]

1. When a DOMAIN_NAME is specified the presence of the name will be checked and processing will halt if the name is invalid.

2. If DOMAIN_NAME is not specified the DOMAIN_NO for the ATTRIBUTE_CLASS will be zero (undefined).

3. All DOMAINS must be created using the CREATE DOMAIN command.

[keyword KEYWORD ...]

1. The keyword will be created in the Common Data Model if it does not already exist.

2. Keyword references are created for the attribute class.

Examples:

```
create attribute class ORDER_DATE domain DATE
        keyword KEY_DATE;
```

CREATE CATEGORY

Syntax:

```
create {complete   } category {relation}  RC_NAME
       {incomplete}
        of EC_NAME
        discriminated by AC_NAME
        into {category EC_NAME1 if 'literal'
             [set {NEW_TAG_NAME= OLD_TAG_NAME}...]}...
      [keyword KEYWORD...] ;
```

Semantics:

1. The following elements must exist in the Common Data Model:
   model
   generic entity (EC_NAME)
   category entities (EC_NAME1...)
   discriminating attribute in generic entity (AC_NAME) primary key in generic entity.

```
create {complete   } category [relation] RC_NAME
       {incomplete}
       of EC_NAME
       discriminated by AC_NAME
```

1. A new categorization relationship is added to model.

2. A complete category relation requires at least two category members.

```
into {category EC_NAME1} if 'literal'
```

1. The primary key of the generic entity will be migrated as the primary key of the category entities.

2. Category entities must have no keys before becoming part of the category relation.

3. The literal value must be unique for each of the category entities in the category relation.

4. The literal value must be a valid specific value in the domain of the discriminator.

```
[set  {NEW_TAG_NAME=OLD_TAG_NAME} ...]} ...
```

1. If the set phrase is specified, OLD_TAG_NAME (the generic entity's tag name) is migrated with the new name of NEW_TAG_NAME (the category entity's tag name.)

2. The set phrase is associated with the previous category entity and not all of the category entities in the command.

```
[keyword KEYWORD...] ;
```

1. The keyword is created if it does not exist.

2. A relation keyword is referenced and created.

Example:

```
create incomplete category EMPLOYEE_TYPES of EMPLOYEE
discriminated by EMPLOYER_TYPE
into category PART_TIME if 'PT'
     category FUL_TIME if 'FT'
keyword EMPLOYEES;
```

CREATE DESCRIPTION TYPE

Syntax:

```
create description type DESC_TYPE...;
```

Semantics:

1. This command will store new, legal description types in the CDM.

2. The DESC_TYPE will be converted to upper case if entered in lower case.

3. The following description types are predefined and stored in the CDM to facilitate description of redundant data:

   DEFINITION
   REPLICATION
   SYNCHRONIZATION
   CURRENCY
   RESPONSIBILITY
   REDUNDANCY

Examples:

```
create description type DEFINITION POLICY USAGE MAINTAINER ;
```

CREATE DOMAIN

Syntax:

```
create domain DOMAIN_NAME standard
   [data] type DATA_TYPE_NAME nddl_data_type
                integer1[:integer2]

[[data] type DATA_TYPE_NAME nddl_data_type integer1[:
              integer2]] ...

[value ('SPECIFIC_VALUE') ...]

[range ('BEGIN_VALUE' thru 'ENDING_VALUE') ...];
```

Semantics:

1. A new domain is created within the system.

2. All data types are added for the domain just created.

3. A standard data type must be specified first. Standard data types must be character, varchar, signed, unsigned, decimal or numeric.

4. The second and all following data types for the domain will be user defined data types.

5. See Appendix A for a list of valid data types.

6. When specifying the maximum size of a data type, integer2 cannot be greater than integer1.

7. SPECIFIC_VALUE, BEGIN_VALUE and ENDING_VALUE must conform to the standard data type's definition. These values and ranges are associated with the standard data type, not with the user data types.

Examples:

```
create domain ZIP_CODE standard
    type ABC character 30
    type XY2 integer 6
    value 'AAAAA'    '123ABC'
       range 'AA' thru '22'
   '1' thru '10';
```

CREATE ENTITY

Syntax:

```
create entity [class] EC_NAME
     [{primary  }[key [class] KC_NAME[= AC_NAME ...]]...
      {alternate}
     [[owned] attribute [class] AC_NAME ...]
     [keyword KEYWORD ...];
```

Semantics:

create entity [class] EC_NAME

1. To complete the above command the following element must exist in the Common Data Model:

   attribute
   create entity [class] EC_NAME

1. A new entity class is created for the current model.

[{primary } key [class] KC_NAME [= AC_NAME ...]]...
 {alternate}

1. A new occurrence of key if created for the entity using KC_NAME.

2. A new occurrence of key member is created for the entity for each AC_NAME. If AC_NAME is omitted, a key member occurrence is created using KC_NAME.

3. There may be only one primary key.

4. If primary or alternate is not specified, the first key will be the primary key and the following keys will be alternate keys.

5. If any key exists for an entity, it must be a primary key.

[owned] attribute [class] AC_NAME ...

1. A new occurrence of attribute use and owned attribute is created for AC_NAME if one does not already exist. AC_NAME can not be owned by any other entity.

2. New attributes for the entity being created are added.

3. Each attribute is created as an owned attribute for the entity.

4. A new occurrence of an attribute use is created.

keyword KEYWORD...

1. Keyword references are created for the entity.

2. The keyword will be created in the Common Data Model if it does not already exist.

Examples:

```
create entity SALES_PERSON
    key class ORDER = ORDER_NUMBER
    owned attribute SALES
    keyword SALES_ID;
```

CREATE MAP

Syntax:

for entity:

```
create map EC_NAME
     to record (DB_NAME.REC_NAME)...

     [(allow   )    retrieval]    [(allow   )    update];
     [(disallow)                   ]    [(disallow)          ]
```

for attribute use:

```
create map EC_NAME.TAG_NAME

[              (replication    )]
[              (relational     )]
[(active )     (original source)]    [for preference PREF_NUMB]
[(passive)     (copy           )]
[              (redundant      )]
[              (backup         )]

               (   [to  field  (DB_NAME.REC_NAME.FIELD_NAME )
                    ...]

[to  set  (DB_NAME.SET_NAME value 'STRING')...]);
```

for relation:

```
create map  EC_NAME1  RC_NAME [EC_NAME2]

     [to  set  (DB_NAME.SET_NAME[.MEMBER_REC_NAME] ) ...]
     [to record DB_NAME.REC_NAME as outerjoin] ;
```

Semantics:

1. This command shall be used to store the mapping definition of an entity or a relation class to the internal schema or the single preference of an attribute use class.

   To complete the above command, the following elements must exist in the CDM:

   a. Integrated_Model (MODEL_NO = 1)

   b. The tag to be mapped must be part of the Integrated_Model.

   c. The relation to be mapped must be part of the Integrated_Model.

   d. The entity to be mapped must be part of the Integrated_Model.

   e. The database, record, set and field being mapped.

f. A horizontal partition must be defined if the same tag is being mapped multiple times for the same preference.

2. When an attribute use (AUC) maps to a data field:

   a. The AUC must not have been previously mapped for this preference number unless the entity has been horizontally partitioned.

   b. To ensure there is always a primary mapping for an AUC, the first mapping to be created for an AUC must be Preference Number 1.

3. When an AUC maps to a set:

   a. At least two sets must be specified.

   b. Each must be single member sets.

   c. The sets may not be used for relation mappings or a mapping from some other AUC.

   d. Each set must be from the same data base and have the same owner record type when the entity is not horizontally partitioned. If the entity is horizontally partitioned, the sets may specify a different record type/database for each partitioned record of the entity.

   e. Each value must be unique; i.e. value 'A' cannot be used for two sets. The value can be re-specified for:

      1) A different preference mapping of the same AUC.

      2) Distinct partition of the entity.

4. When a Relation maps to a set:

   a. The MEMBER_RECORD_NAME may be omitted if the set to be mapped to is a single member record type set; otherwise the member record name is required.

   b. There must be no previous mappings to the set.

   c. Link relation requires independent_entity_name, rc_name and dependent_entity_name. Category relations require generic entity name and rc_name.

5. When a relation maps to a record as outerjoin:

   a. This clause allows the CDM Administrator to map two related entities to a single internal record.

   b. The cardinality of the relation may be 0:many or 0:1. The cardinality may not be 1:many since this implies a Join.

c. A 0:many cardinality implies that the internal record represents the child entity; a 0:1 cardinality implies that the internal record represents the parent entity.

d. Before defining a record outerjoin mappping, the entity to record mapping must exist.

e. A relation may not map to both a set and a record in an outerjoin structure.

f. It is recommended that all attributes in an outerjoin structure be mapped to the record for the same preference number.

g. Link relation requires independent_entity_name, rc_name, and dependent_entity_name. Category relation requires generic_entity_name and rc_name.

6. A Relation to data field map is illegal.

7. When an Entity maps to a record:

   a. The user must specify all record types that the entity may "map to" for any preference at the attribute level and for any horizontal or vertical partitions.

   b. The entity must be mapped to a record before an attribute of the entity is mapped to a data field of the record.

   c. The CDM Administrator must choose to ALLOW or DISALLOW UPDATE programs to be generated for any potential secondary copies (preference number > 1 and < 51). The default is to DISALLOW UPDATE.

   d. The CDM Administrator must choose to ALLOW or DISALLOW RETRIEVAL programs to be generated that access potential secondary copies (preference number > 1 and < 51) on the same host on which the precompiled application executes. The default is DISALLOW RETRIEVAL.

8. Use the Define Algorithm command to map an AUC through a complex mapping algorithm. Define Algorithm defines the use of the software module as a complex mapping algorithm.

9. Preference numbers do not apply to relation mappings.

10. Preference number, unless specified, defaults to 1 for an ACTIVE mapping. The preference number defaults to 51 for a PASSIVE mapping.

11. PASSIVE mappings will be ignored by the NDML precompiler. To preserve upward compatibility, ACTIVE mappings are the default.

12. The following mapping classifications are allowed:

REPLICATION	COPY
RELATIONAL	REDUNDANT
ORIGINAL SOURCE	BACKUP

ORIGINAL SOURCE is the default if preference number = 1, REPLICATION is the default for preference numbers 2 through 50, and REDUNDANT is the default for preference numbers 51 through 99.

Examples:

```
create map PART.SIZE for preference 1
      to field PARTDB.PREC.PSIZE;
create map PART.SIZE for preference 2
      to field PARTDB.PREC2.PSIZE2·

create map ORDER.STATUS for preference 1
      to set CUSTDB.INPROCESS value '1'
             CUSTBD.BAKORDER  value '2'
             CUSTDB.SHIPPED   value '3' ;

create map EMPLOYEE POSSESSES SKILLS
      to set SKILLDB.SETA.REC4
             SKILLDB.SETB  ;

create map EMPLOYEE
      to record SKILLD.EMP-REC
             allow retrieval  allow update;

create map PART.SIZE active original source for preference 1
      to field PARTDB.PREC.PSIZE;


create map PART.SIZE ACTIVE REPLICATION for preference 2
      to field PARTDB.REC.PSIZE2;

create map PART.SIZE passive copy
      to field PC.LOTU527.COL32;

create map EMPLOYEE HAS INSURANCE
      to record TEST.PERSONNEL as outerjoin;
```

Syntax:

```
create model MODEL_NAME;
```

Semantics:

create model MODEL_NAME

1. A new model is created as unchecked in the system.
2. The model remains the "current" model for all other modeling commands until another CREATE MODEL or ALTER MODEL command is entered.

Examples:

```
create model DEF_COMPANY;
```

CREATE PARTITION

Syntax:

```
create partition [INTEGER1] of entity  EC_NAME
      to record
                    DB_NAME.REC_NAME ...;
```

Semantics:

1. This command shall be used to define a horizontally partitioned entity of the INTEGRATED_MODEL.

2. EC_NAME must exist in the INTEGRATED_MODEL (conceptual schema).

3. The records named must be previously defined to the CDM's internal schema.

4. Multiple horizontal partitions are allowed for an entity; they are distinguished by the user assigned INTEGER1.

5. If INTEGER1 is not specified, a "1" will be assumed.

6. There must be at least two DB_NAME.REC_NAME partitions specified for an entity.

7. The same record type must not participate in more than 1 partition of the same entity class.

8. At this time, update of entities with horizontal partitions are not supported by the NDML precompiler.

Examples:

```
create partition 1 of entity ORDERS
       to record

                    DB1.ORDERS
                    DB2.ORDERS
                    DB3.ORDERS;

create partition 2 of entity ORDERS
       to record

                    DB1.CUSTORD
                    DB2.CUSTORD  ;
```

Syntax:

```
create relation [class] [{INTEGER1}]
                        {many    }
  EC_NAME| RC_NAME [INTEGER2: {INTEGER3}] EC_NAME2
                             {many    }
   [migrates  KC_NAME
   [set (NEW_TAG_NAME = OLD_TAG_NAME)...]]
          [keyword KEYWORD ...];
```

Semantics:

```
create relation [class] [{integer1|many}] EC_NAME1
          RC_NAME [{integer2: integer3|many}] EC_NAME2
```

1. The following elements must exist in the Common Data Model:

   model
   independent entity (EC_NAME1)
   dependent entity (EC_NAME2)

2. A new link relation is inserted.

3. Cardinalities for the entities of the relation are inserted. If integer1 or many is omitted, the independent cardinality defaults to one. If integer2 is omitted, the left dependent cardinality defaults to zero. If integer3 or many is omitted, the right dependent cardinality defaults to many.

4. Integer3 may not be less than integer2.

5. Integer3 may not be zero.

```
migrates KC_NAME [set (NEW_TAG_NAME = OLD_TAG_NAME) ...]
```

1. If this phrase is not present and there exist a primary key for the independent entity, the primary key will be migrated as if the phrase was specified.

2. The following elements must exist in the Common Data Model:

   key for independent entity
   key members for independent entity
   attribute use class for each key member - associated with the independent entity

3. The key cannot have been previously migrated.

4. The dependent entity cannot end up with attribute use with the same tag name.

5. An attribute use and an inherited attribute use for the dependent entity are created for each key class member of the independent entity migrated to the dependent ENTITY.

6. If the set phrase is specified, OLD_TAG_NAME (the independent entity's tag name) is migrated with the new name of NEW_TAG_NAME (the dependent entity's tag name).

7. A complete relation occurrence is created.

keyword KEYWORD ...

1. The keyword is created if it does not exist.

2. A relation keyword reference is created.

Examples:

```
create relation INVOICE SUPPLIES ORDER_INFO
        migrates ORDER set ORDER_NUMBER = ORDNO
        keyword ORDER_DATA;
```

CREATE UNION

Syntax:

```
create union of record DB_NAME.REC_NAME
  to entity  {EC_NAME when FIELD_NAME op 'STRING'}
             {[AND FIELD_NAME op 'STRING']...} ...   ;
```

Semantics:

1. All entities that map to the same record type as a union mapping must previously be defined in the CDM.

2. Each entity of the union must have at least one, and perhaps many, union discriminator conditions. The data fields must be defined for the union record.

3. The entity must be part of the INTEGRATED_MODEL.

4. For the word op above in the union discriminator clause, the user must use the following: =, <, >, >=, <=, !=

Examples:

```
create union of record   DB2.RECX
    to entity   E1 when  DF1 = 'A'
                E2 when  DF1 = 'B' and DF2='2'
                E3 when  DF1 = 'B' and DF2='1' ;
```

CREATE VIEW

SYNTAX

```
create view VIEW_NAME

  [data item [DATA_ITEM_NAME [data] type DATA_TYPE_NAME]...]

as select [distinct]          { *                          }
                              { TAG_NAME                  }...
                              { [ABBREV.]TAG_NAME         }...
  from
                              {EC_NAME              }
                              {EC_NAME.ABBREV             }...

  [where

    [ {[{ABBREV.}] TAG_NAME  join_op  [{ABBREV.}]  TAG_NAME
    [AND]}...] ]

|_{                                                    }          _|
| {        {[{ABBREV.}] TAG_NAME  op {'string'} )     }           |
| {        {                          ( number ) )    }           |
| {                                                   } [{AND}]   |
| {[[(][NOT]]{[{ABBREV.}]TAG_NAME is [not] null} [)]} } [{OR}]..  |
| {                                                   } [{XOR}]   |
| {        {[{ABBREV.}]TAG_NAME  [not] between        }           |
| {        {             {'string' and 'string'} }    }           |
| {        {             { number  and  number } }    }           |
|_{                                                   }          _|

 ]   ;

 where join op in ( =   ,  U=)
 where      op in ( =, >, >=, <, <=, !=, like)
where number is integer or real
```

SEMANTICS

1. To create a VIEW using the above command the following elements must exist in the Common Data Model (CDM).

   - entities named
   - data items and attribute uses from the same domain
   - existence of standard data type
   - link or category relations with migrated keys between the entities named

```
create view VIEW_NAME
```

1. A VIEW is built using one entity or more than one entity of the INTEGRATED_MODEL. It allows the creator to automatically qualify certain rows based upon values in the WHERE clause. A view is used to query specified attributes from a specified entity or entities.

2. A USER_VIEW occurrence is created if it did not previously exist.

data_item DATA_ITEM_NAME [[data] type DATA_TYPE_NAME]...

1. This clause must be specified when views are built from more than one entity class.

2. The data items must correspond in order and number to the attribute use (TAG_NAME) specified in the select clause.

3. If the data type name is not specified, the data type will become the standard data type for the corresponding attribute use. If the data type name is specified, a check is made to ensure compatibility between the data item and attribute's standard data type name.

4. When data items are specified for a view built from multiple entities, the SELECT * clause may not be used.

```
as select [distinct]
          { *  }
          { TAG_NAME     }
          { [ABBREV.]TAG_NAME...] }
```

1. If an * is specified:

   a. the FROM clause may not be omitted.
   b. a DATA_ITEM and PROJECT_DATA_ITEM occurrence is created for each attribute use associated with the entity named on the FROM clause.

2. If the tags are being selected from a single entity, the ABBREV may be omitted provided a FROM clause is specified without an abbreviation.

3. If tags are being selected from more than one entity, both abbreviations and tag names must be specified and a FROM clause with entity names and abbreviations must be specified.

4. DATA_ITEM and PROJECT_DATA_ITEM occurrences are created for each attribute use named in the select.

5. If abbreviations are used, they must be used for all attribute uses.

6. The optional distinct clause may be specified to prevent duplicate rows being returned to the user.

7. The ABBREV has a two character maximum.

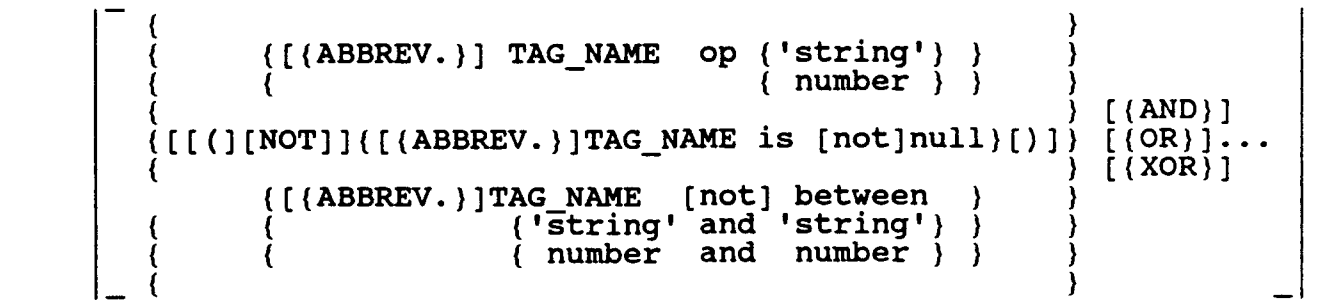
```
from {EC_NAME        }
     {EC_NAME.ABBREV}...
```

1. The abbreviation for the entity must be included if the ABBREV is used in the select clause or where clause.

2. A VIEW_EC_XREF occurrence is created for each entity named in the FROM clause.

```
[where

[{[{ABBREV.}] TAG_NAME join_op [{ABBREV.}] TAG_NAME  [AND]}...]]
```




```
 ]
```

1. The WHERE clause must be specified when tags are being selected from more than one entity. Abbreviations must be used in place of the entity names in the WHERE clause. The abbreviation must be the same as the abbreviation used in the FROM clause. If the view consists of a single entity, the ABBREV may be omitted provided a FROM clause was specified without an abbreviation.

2. The entities specified in the FROM clause must form a confluent hierarchy, with each entity appearing as either an independent, or generic, category, or dependent entity in the hierarchy.

3. A complete relation must exist between each independent or generic and dependent or category entity in the view.

4. Join conditions (abbrev. tag_name op abbrev. tag_name) between each independent or generic and dependent or category entity in the view must be specified, using at least one key member from the independent or generic entity. Non-key attributes from the independent or generic entity may not be used as join criteria. Join conditions must be ANDed together. Legal join operators are equi-join (=) and outer-join (u=).

5. Additional qualification criteria (abbrev. tag_name op value) may be specified to restrict the rows a user can access. The restriction can consist of a single search condition or a series of search conditions linked by the logical operators AND, OR, XOR. Each condition compares an

attribute (tag) to a literal or constant or NULL. The comparison is done with the operators (<, <=, =, !=, >=, >) or with special functions (NOT, BETWEEN, NOT BETWEEN, IS NULL, IS NOT NULL). Conditions can be nested with PARENTHESES using standard Boolean logic.

6. Additional qualifications can be made using the 'like' operator. This compares an attribute (tag) to a string.

   a) The "like" operator can only be used with data items that are defined with a data_type of CHAR, CHARACTER, or VARCHAR.

   b) The percent sign (%) is used for 0:many character match.

   c) The underscore (_) is used for a 1:1 character match.

   d) The pattern matching characters may be placed anywhere in the string and used as many times as needed.

Examples:

```
create view EMP
      as select*from EMPLOYEE
           where LAST_NAME like 'SM%'
                and FIRST_NAME like '%J__o';
```

7. The attributes specified in the WHERE clause must exist in an entity specified in the FROM clause.

8. A VIEW_QUALIFY_CRITERIA occurrence is created for each item in the WHERE clause.

9. A VIEW_QUAL_XREF OCCURRENCE IS CREATED FOR EACH attribute in the WHERE clause.

10. Views created with multiple entities are not updatable i.e., NDML INSERT, MODIFY, and DELETE operations will not be performed.

DEFINE ALGORITHM

Syntax:

```
define algorithm MOD_NAME [MOD_INSTANCE] for  (retrieval)
                                              (update   )
     [for preference PREF_NUMB]

     using [parameters]

     ([PARM_ID] (from     ) (dataitem  )
                (constant) (attribute ) (ID_1.ID_2[.ID_3])  )...
                (to       ) (datafield ) ('string'        )  )
                            (record)

     status;
```

Semantics:

1. To complete the above command, the following elements must exist in the CDM:

   a) a software module with its parameter list.

   b) the input or output attributes must be defined.

   c) dataitems or datafields or a record must be defined.

2. This command is used to define the use of a software module as a complex mapping algorithm. For CS/IS mappings this command maps a tag or many tags to a data field or many data fields or a record thru a complex mapping algorithm, for a stated preference. PREF_NUMB if not specified defaults to 1. A software module may be used for many different mapping algorithms, and is distinguished by MOD_INSTANCE. MOD_INSTANCE, if not specified, defaults to 1.

3. The software module name (MOD_NAME) must have been previously defined using a DEFINE MODULE command. Each parameter must be accounted for in the using list.

4. If the keyword "parameters" is specified, PARM_ID must be used on each entry. If the keyword "parameter" is omitted, eac parameter must coincide exactly in order with the parameters defined in the DEFINE MODULE command.

5. Status must be the last parameter. It is an output parameter, and must conform to the IISS error handling philosophy; i.e., it must be a COBOL picture X(5). A successful status is all zeros.

6. The algorithm must be specified for use on NDML retrieval or update requests. "Retrieval" means an IS to CS to ES conversion. Conversion direction "update" means an ES to CS to IS conversion direction.

7. Entries on the using list must begin with "from" for

algorithm inputs. "To" parameters must specify the algorithm outputs. Constants are non-varying input parameters to the algorithm.

8. "From" and "to" parameters must specify the mapping type - dataitem, data field, attribute, or record, followed by the clause ID_1.ID_2[.ID_3]. None of the mapping types are allowed with "constant". The following option must be a 'STRING' indicating the value to be used.

9. The use of ID_1, ID_2 and ID_3 are defined as follows:

	ID_1	ID_2	ID_3
dataitem	VIEW_NAME	DATA_ITEM_NAME	X
attribute	EC_NAME	TAG_NAME	X
record	DB_NAME	REC_NAME	X
datafield	DB_NAME	REC_NAME	FIELD_NAME

10. For external/conceptual mappings, data item and attribute mapping types are involved. A single "from" parameter and a single "to" parameter can be specified, along with any number of constants.

11. For conceptual/internal mappings, attribute, record and datafield mapping types are involved. Many "to" and "from" datafields and attributes may be specified but only one record is allowed as input or output to the algorithm.

12. The rules below must be followed when defining algorithms:

    Retrieval rules:

    - AUCs must be from related entity classes
    - datafields must belong to the same Record type

    Update/search rules:

    - AUCs must be from the same entity class
    - datafields must belong to the same Record type

    This matrix describes the legitimate input/output parameters for an algorithm (where I is Input to the algorithm an O output):

	RETRIEVAL	SEARCH/UPDATE
ES to CS	X	dataitem (I) attribute (O)
CS to IS	X	attribute(s) (I) datafield(s) (O) record (O)
IS to CS	datafield(s) (I) record (I) attribute(s) (O)	X
CS to ES	attribute (I) dataitem (O)	X

13. Users should be aware of the union mapping capability of the Create Union command; user should not define CMAs when a record union should be used instead.

Examples:

```
define      algorithm     X52     for         update
      using parameters
            IN_REC        from    dataitem    VIEW_1.ONE_ITEM
            METRIC_CONV           constant    '2.54'
            PART_WEIGHT    to     attribute   EC1.TAG1
            STATUS;

define      algorithm     MOD1    for         retrieval
      using parameters
            PARM1         from    datafield   DB1.REC1.DF1
            PARM2         from    datafield   DB1.REC1.DF2
            PARM3         to      attribute   EC1.TAG1
            PARM3         to      attribute   EC2.TAG2
            STATUS;
```

Syntax:

```
define DBMS_NAME   {database}   named DB_NAME on host HOST_ID
                   {pcb     }

    [with password  PW_ID ]
    [schema S_NAME and subschema SS_NAME areas AREA_NAME...]
    [located at 'STRING']

    [position INTEGER1 in psb PSB_NAME feedback length INTEGER2]

    [[stores] character null [as]   {zeros        }]
                                    {spaces       }
                                    {all 'STRING2'}
                                    {null         }

    [[stores] integer null [as]     {zeros        }]
                                    {low-values   }
                                    {high-values  }
                                    {null         }

    [NTM directory 'STRING3']   ;
```

Semantics:

1. The database name, DB_NAME, must be unique.
2. DBMS_NAME must be previously defined, using the DEFINE DBMS command.
3. HOST_ID must be defined, using the DEFINE HOST command.
4. An association must exist between the DBMS and HOST.
5. If character null clause or integer null clause is omitted, null value defaults to 'null' for a relational DBMS and 'zeros' for a non-relational DBMS.
6. NTM directory clause if omitted, defaults to "GR".
7. The database created is established as the current database until such time the user defines another database.
8. No special clauses are required for a DB2 or TOTAL database.

Oracle Database

```
define ORACLE database named DB_NAME on host  HOST_ID

    with password  PW_ID;
```

1. A password is required.

VAX-11/IDMS/IDS-II database

```
define VAX-11 database named DB_NAME on host  HOST_ID

    with schema S_NAME and subschema SS_NAME areas AREA_NAME...
         [located at 'STRING']
```

1. The schema, subschema and at least one area is required.

2. The located at phrase is required. It is used to specify the full directory name in which a VAX-11 database is stored.

IMS database

```
define IMS pcb named DB_NAME on host  HOST_ID

    with position INTEGER1 in psb  PSB_NAME
         feedback length INTEGER2;
```

1. An IMS database requires the term 'pcb'

2. The start position and feedback length are also necessary.

3. The PSB_NAME must previously exist and be stored with the DEFINE PSB command.

Examples:

```
define ORACLE database names ORC_DB on host VAX with password
SOS;

define VAX-11 database named COD_DB on host VAX schema S1 and
subschema SS1 areas A1 A2;

define IMS pcb named IMS_DB on host IBM with position 1 in psb
feedback length 20;

define DB2 database named DB2_DB on host IBM
stores character null as all '??'
    integer null as null
NTM directory 'GD';
```

Syntax:

```
define dbms DBMS_NAME model  DBMS_TYPE
    [on host HOST_ID]...    ;
```

Semantics:

1. DBMS_TYPE must be one of the following:

   H - hierarchic
   R - relational
   N - network
   I - indexed
   S - sequential

2. A HOST must be defined to the CDM before a cross reference can be made to a DBMS.

3. DBMS_NAME must not already be defined.

Examples:

```
define dbms INRGES  model  R
      on host NCR;
define dbms S2K  model H ;
```

Syntax:

```
define host HOST_ID
      [with dbms   DBMS-NAME]...;
```

Semantics:

1. HOST_ID must not already be defined.

2. The DBMS must be defined to the CDM before a cross reference can be made to a HOST.

Examples:

```
define host H12
      with dbms IDS-II;

define host HP
      with dbms RIM ORACLE;
```

Syntax:

```
define module MOD_NAME in LANGUAGE
      [parameters {PARM_ID [data] type DATA_TYPE_NAME}...];
```

Semantics:

1. This command defines a software module to be used for complex mapping algorithms. Parameters for the routine must be defined in the order the subroutine expects them. Since, this can be used to define any other software modules in the system, the parameter list may be empty.

2. MOD_NAME must be unique in the system.

3. At this time, the values for LANGUAGE are not controlled.

4. PARM_ID is the name of the parameter.

5. DATA_TYPE_NAME refers to the user defined data type describing the parameter.

6. If the module is defined for use as a complex mapping algorithm, the last parameter should be the output parameter to check error codes, and should conform to the IISS error handling philosophy (COBOL PIC X(5)).

Examples:

```
define module X52 in COBOL
      parameters

           IN_REC          type      PART_RECORD
           PART_WEIGHT     type      KILOS
           STD_STATUS      type      RET_STATUS ;
```

DEFINE PSB

Syntax:

```
define psb PSB_NAME on host HOST_ID ;
```

Semantics:

1. The PSB_NAME must not be previously defined to the CDM.
2. The HOST_ID must be previously defined.

Examples:

```
define psb ABCD on host I21;
```

DEFINE RECORD

Syntax:

```
define  {table  }    REC_NAME  [of   {database}   DB_NAME  ]
        {record}                     {pcb     }
        {segment}

      [ in areas AREA_NAME ... ]

          with     {fields  }
                   {columns }
                   {elements}
                   {items   }

  (   [ LEVEL_NO ]          {FIELD_NAME}
                              {filler}

      [    {[data]   type DATA_TYPE_NAME}]
           {         FILLER_SIZE     }

      [ occurs INTEGER1 [depending on FIELD_NAME_2]]

      [ indexed by FIELD_NAME_1 ]]

      [ redefines FIELD_NAME_3 ]

      [   {known   }  ]
          {unknown}

      [ [ {unique } ] key ]          ) ... ;
        [ {duplicate}]
```

Semantics:

1. This command defines a table/record/segment for a previously defined database.

2. The "of database/pcb DB_NAME" option can be omitted if a current database is established in the session.

3. Records must be defined in the exact same order as found on the dbms definition, or the actual record description as used by existing software.

4. "Filler" and an integer FILLER_SIZE can be used to "hide" information or data areas not considered common data. NDML updates to records containing fillers are not supported, i.e. the filler contents are not guaranteed. Any number of "filler" entries can be used in a record description. Filler entries should be used for unassigned areas or fields which are not common data known to the CDM. The entire record layout must be defined. If the order of data fields is not important to the dbms, e.g. ORACLE, then the order in NDDL is not important.

5. If OCCURS clause is omitted, then it is assumed the field occurs one time. If the OCCURS clause is specified, INTEGER1 should be greater than 1. If the "depending on" field is specified along with the OCCURS clause, FIELD_NAME_2 must be defined earlier in the record layout. If the "indexed_by" field is specified along with the OCCURS clause, FIELD_NAME_1 must be defined earlier in the record layout. If FIELD_NAME_1 is not defined, FIELD_NAME_1 will be created by the NDDL command processor with a default datatype of INDEX_TYPE, and specified as UNKNOWN to the DBMS.

6. A field can be specified as a key. This implies the DBMS can use it for direct access. A key can be a unique key index or a duplicates allowed key index. If neither is specified, but "key" is specified, a unique key index is assumed.

7. Indexed by FIELD_NAME_1 indicates the field has a COBOL index. A conceptual attribute may map to the index itself. FIELD_NAME_1 cannot have its own index or index more than one data field.

8. Depending on FIELD_NAME_2 is used to describe the COBOL capability of occurs depending on variable repeating groups. A repeating field or repeating group cannot be a key.

9. Redefines FIELD_NAME_3 allows the field to redefine all or part of another data field. The redefining can also be a component (or group) data field. FIELD_NAME_3 must be a unique key if it redefines a field that is a unique key.

10. The user should indicate if the field is known or unknown to the dbms, i.e. can the dbms address this field by name. If left off, the default is known. Note, records can be defined with no known data fields if the dbms supports entire record access. A group data field should be specified as known to the DBMS when its sub-components are specified as unknown and vice versa.

11. LEVEL_NO clause, if not specified, defaults to 1. Each subcomponent data field must use the LEVEL_NO clause. Levels must be incremented by one. Any level of subcomponent structures can be represented. A subcomponent of a key may be key too. The restriction is that subcomponent of a key can only be a secondary key or duplicate key.

12. An index field cannot redefine or be a component, group or a key.

13. No clauses apply to a filler, only the FILLER_SIZE is needed.

14. There is no restriction on the number of unique or duplicate keys defined for a record.

15. An elementary item must have a data type name that is predefined in the CDM table USER_DEF_DATA_TYPE.

16. The data-type of a field must be a valid option for the related DBMS.

    Relational DBMS:
    (ORACLE, DB2, INGRES5, INGRES6)

    C- Character
    P- Packed
    I- Integer
    V- Variable length Character
    A- Small Integer

    CODASYL DBMS:
    (IDMS, VAX-11)

    C- Character
    P- Packed
    I- Integer
    F- Float
    V- VAriable length Character
    A- Small Integer
    O- Double Precision
    N- Unsigned

ORACLE Records

```
define table REC_NAME [of database DB_NAME]
      with columns
      FIELD [[data] type DATA_TYPE_NAME]

         [   known     ]
             unknown       ;
```

1. DATA_TYPE_NAME should be specified since ORACLE supports only elementary datafields.

2. ORACLE does not support repeating fields, COBOL type indexes, or redefinition. The CDM does not need knowledge of ORACLE's indexes (keys).

3. If an ORACLE column is not COMMON DATA, do not use FILLER, but simply exclude the column definition.

4. ORACLE field must be 01 level and must be defined as "known" to the DBMS.

5. Data_Types of Integer or Small Integer must have a size of 4 or 9 only.

TOTAL Records

1. All clauses are pertinent to TOTAL since the application may redefine the entire record in any way it sees fit.

2. Variable records should not have keys defined.

3. All redefinitions of TOTAL variable records should be specified to the CDM as a single field.

4. TOTAL only allows a single unique key for MASTER records only.

5. The record name must be the TOTAL 4 character file name.

6. To define a TOTAL file_name, the REC_NAME will be truncated, left justified to 4 positions and placed in the CDM tables DATA_BASE_AREA and DB_AREA_ASSIGNMENT.

CODASYL Records

1. All clauses are pertinent to CODASYL data records because access is done on a record at a time basis.

IMS Records

1. All clauses are pertinent to IMS data records because access is done on a segment at a time basis.

Examples:

for ORACLE

```
define table T1 of database ORC_DB with columns
      C1     type     CHARACTER_NAME     known
      C2     type     CHARACTER_NAME     known
      C3     type     CHARACTER_NAME     known
      C4     type     CHARACTER_NAME     known    ;
```

for CODASYL

The following examples show how to translate from a record layout:

```
01  REC6
    03    PART-JOB-GROUP.
          05  JOB-STATUS            PIC X(8).
          05  JOB-INFO   REDEFINES      JOB_STATUS.
              07  JOB-ID            PIC XX.
              07  JOB-DATE          PIC X(6).
    03  PART-CUST-GROUP OCCURS 100 TIMES INDEXED BY C-INDEX.
          05  CUST-NAME             PIC X(30).
          05  CUST-ADDR             PIC X(50)
    .
    .
    .
define record REC6 of database DB1

    with fields
  1       C-INDEX              type INDEX_TYPE unknown
  1       PART-JOB-GROUP       unknown
    2          JOB-STATUS      type CODE-DATE-TYPE
    2          JOB-INFO        redefines JOB-STATUS
      3   JOB-ID    type CODE-TYPE
      3   JOB-DATE  type DATE-TYPE
  1       PART-CUST-GROUP      indexed by C-INDEX occurs 100
                                 unknown
    2           CUST-NAME      type NAME-TYPE
```

```
    2           CUST-ADDR     type ADDR-TYPE  ;
```

for IMS

```
    define segment S1 of pcb I1 with elements
          X2 type D4
          X5 type D5 unknown
          filler 5  ;
```

for TOTAL

```
    define record VAR1 with elements

       1 CODE             type       TOT-REC-CODE
       1 KEY1             type            X10
       1 KEY2             type            X20
       1 filler           type            X4
       1 DATA             type            X100
       1 RECA             redefines       DATA
          2 AAA1          type            X50
          2 AAA2          type            X20
          2 filler        30
       1 RECB             redefines       DATA
          2 BBB1          type            X100
             3 B1AA       occurs 10       unknown
                4 B1AB    type       X10  unknown ;
```

Syntax:

```
define  {set} [SETNAME]  of  {database}  DATA_BASE_NAME
        {path}               {pcb}

{relating} RECORD_ID1 to {RECORD_ID2 [required|optional]}...
{from}

            [linked by  DATA_FIELD_NAME]    ;
```

Semantics:

1. The above command is illegal for a relational DBMS data base.

2. To complete the above internal schema command the following elements must exist in the Common Data Model:

   data base
   record type
   data field

3. The following rules govern the creation of a set/path for DBMS type:

   a. Sets may be created for TOTAL, IMS (called paths), and the CODASYL DBMS's: VAX-11, IDMS, and IDS.
   b. CODASYL DBMS's require an entry for required/optional members.
   c. TOTAL DBMS's require a linked by Data Field Name clause.
   d. Single members are allowed in all DBMS's. Multiple members (Record_List_ID) may be used for the CODASYL DBMS's.

```
define   {set} [SETNAME]  of   {database}   DATA_BASE_NAME
         {path}                {pcb}
```

1. The set or path is a required entry for TOTAL and CODASYL DBMS's. Path is used as a more natural entry for an IMS data base. For an IMS database, if "path" is used the set name is derived from combining the owner (RECORD_ID1) and member (RECORD_ID2) names.

2. The DATA_BASE_NAME or pcb is an optional entry if a DATA_BASE_NAME or pcb has previously been established during the session using a DEFINE DATABASE, DEFINE RECORD, or DEFINE SET command. The use of pcb is more natural for an IMS data base.

```
{relating}   RECORD_ID1  to {RECORD_ID2 [required|optional]}...
```

{from}

1. The relating/from clause is a required entry. From is more natural with an IMS data base. RECORD_ID1 denotes the owner of the set and RECORD_ID2 denotes the member(s) of the set. RECORD_ID2 is used for CODASYL DBMS's to show multiple members and and includes the member RECORD_IDs separated by a space. The required or optional clause may only be used for the CODASYL DBMS's. All other DBMS's default to required.

[linked by DATA_FIELD_NAME]

1. The linked by clause may only be used for a TOTAL DBMS and must include a DATA_FIELD_NAME from the variable record (RECORD_ID2).

Examples:

```
define set MAY_HAVE of database DB1 relating CUSTOMER_REC to
CUSTOMER_ACTIVITY ;

define path of pcb DBIMS2 from SUPPLIER_ACCT to
SUPPLIER_INVOICE ;

define set CONTROLLED_BY of database  DBTOTAL14 relating
SHOP_SUPPLY to SHOP_REQUESTS linked by SHOP_NO ;
```

Syntax:

```
describe DESC_TYPE [of] OBJECT_IDENTIFIER [class]
OBJECT_NAME

      [ "description text"
      |  from 'FILENAME.EXT'  ]  ;
```

Semantics:

1. Following is a list of objects that may be described and must exist in the Common Data Model:

OBJECT TYPE	OBJECT NAME(S)
attribute	AC_NAME
entity	EC_NAME
relation	EC_NAME1 RC_NAME [EC_NAME2]
database	DB_NAME
record	DB_NAME REC_NAME
datafield	DB_NAME REC_NAME FIELD_NAME
set	DB_NAME SET_NAME
dataitem	VIEW_NAME DI_NAME
view	VIEW_NAME
userdatatype	DOMAIN_NAME DATA_TYPE_NAME
domain	DOMAIN_NAME
keyword	KWORD_NAME
model	MODEL_NAME
host	HOST_NAME

2. The description types DESC_TYPE must previously exist. It can be stored using the CREATE DESCRIPTION command.

3. All new descriptive text will replace any pre-existing descriptive text for a particular description type and object.

4. No embedded quotation marks may be entered in the description text string.

5. There are three sources of descriptive text; the command itself, from a named file or through the forms text editor.

   a. To enter descriptive text from the command itself use the "descriptive text" option.

   b. When using the file "from FILENAME.EXT" option each record may be no longer than 80 characters. There is no practical limit on the number of records. Each quotation mark in the descriptive text from the file will be replaced by an apostrophe.

   c. To enter descriptive text from the forms text editor consult the IISS UI/VTI Users Manual for Configuration Item: Text Editor.

6. To delete descriptive text from the model, use the describe command with an empty descriptive text string: describe SOURCE of ENTITY E3 "";

7. The current model must be established (ALTER MODEL) to describe attributes, entities and relations.

8. If object type is relation, EC_NAME1 RC_NAME EC_NAME2 signifies a link relation and EC_NAME1 RC_NAME signifies a category relation.

Examples:

```
describe DEFINITION of ATTRIBUTE CUST_ADDR
  "This is the customer bill-to-address. " ;

describe DESCRIPTIVE_NAME ENTITY EMP_INFO
  from 'EMPINFO.TXT' ;

describe SOURCE of ENTITY EMPL "" ;
```

To enter descriptive text from the text editor:

```
describe SOURCE of ENTITY EMPL ;
```

DROP ALGORITHM

Syntax:

```
drop algorithm MOD_NAME [MOD_INSTANCE] for  {retrieval}
                                            {update   }

[for preference    [PREF_NUMB] ;
```

Semantics:

1. All references of the module MOD_NAME for complex mappings will be disassociated.
2. MOD_INSTANCE, if not specified, defaults to 1.
3. The module itself is not dropped.
4. Preference number, if not specified, defaults to 1.
5. If the algorithm being dropped is a conceptual to internal conversion algorithm, its mapping is also deleted.

Syntax:

```
drop alias  { entity EC_NAME
            | attribute AC_NAME } ;
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:
   alias for the entity or
   alias for the attribute

2. The EC_NAME or AC_NAME specified must be an alias for the entity or attribute and it is deleted.

Examples:

```
drop alias entity CUST;
drop alias attribute ORDNO ;
```

DROP ATTRIBUTE

Syntax:

```
drop attribute ATTRIBUTE_NAME ...;
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   model
   attribute

2. The attribute is deleted; if owned, all occurrences of the attribute are removed from owned attribute, attribute use, key member and inherited attribute use.

3. The attribute occurrence is deleted from the model.

4. Those key occurrences which would no longer have any key members are deleted.

5. If a key is deleted the occurrence of a complete relation is also deleted.

6. All keywords associated with the attribute will be dropped.

7. The primary name and all aliases for the attribute will also be deleted.

8. All description texts for the attribute will be deleted.

9. Attributes will not be dropped when mappings exist to data fields, sets, unions, partitions or algorithms.

10. If the attribute being drop is a discriminator in a category relation, the category relation will also be dropped with complete ripple down.

Examples:

```
drop attribute PHONE_NO;
```

Syntax:

drop category [relation] {EC_NAME RC_NAME}...

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   category relation, generic entity, category entities, model

2. The key is "unmigrate" from the category entities and any other entities they have migrated, the attribute use and inherited attributes are removed from these entities.

3. The category relation and complete relation are deleted from the model.

4. The keywords associated with the relation are dropped.

5. All description texts for the relation are dropped.

6. Relations will not be dropped, when category relation to set mappings are present.

Example:

drop category EMPLOYEE EMPLOYEE_TYPES;

DROP DATABASE

Syntax:

```
drop   DBMS_TYPE   {database}   named DB_NAME   ;
                   {pcb      }
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   database/pcb

2. The database is not dropped if the records, sets or fields are mapped to the conceptual schema.

3. This command drops the database, all associated record types, record sets, and data fields.

4. If the records, fields or sets of the databases are mapped to the conceptual schema thru a tag, category or link relation, or complex mapping algorithm or participate in record unions or horizontal partitions, the database will not be dropped.

5. All description texts for all associated sets, fields, or records and the data base will be deleted.

Examples:

```
drop ORACLE database named ORC_DB ;

drop TOTAL database named TOT_DB ;

drop VAX-11 database named COD_DB ;

drop IMS pcb named IMS_DB ;
```

Syntax:

```
drop dbms DBMS_NAME ...;
```

Semantics:

1. DBMS_NAME must previously exist.
2. If the dbms is cross referenced with a host, or any database is associated with the DBMS, the DBMS will not be dropped.

Examples:

```
drop dbms IMS TOTAL ;
```

DROP DESCRIPTION TYPE

Syntax:

```
drop description type DESC_TYPE ... ;
```

Semantics:

1. This command will drop the description types entered. If any descriptions of any objects appear in the CDM, it cannot be dropped.

2. The description type will be converted to upper case if entered in lower case.

Examples:

```
drop description type USAGE MAINTAINER ;
```

Syntax:

```
drop domain DOMAIN_NAME ... ;
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   domain

2. If data types of the domain to be dropped are associated with any data fields, data items or attributes, the usage will be reported and the domain will not be dropped.

3. If condition 2 is not met, the data types associated with the domain are deleted, and then the domain itself is deleted.

4. All descriptive text will be deleted for the domain.

Examples:

```
drop domain ADDRESS;
```

DROP ENTITY

Syntax:

```
drop entity EC_NAME ... ;
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   entity
   model

2. Any owned attribute occurrences are deleted; also removed are attribute use, inherited attribute, key member and keys.

3. All link relations involving the entity are deleted, also any keywords associated with the entity.

4. If a generic entity is dropped, the category relation is dropped with complete ripple down.

5. If a category entity is dropped, the entity will be dropped from the category relation. If category entity is the only member of the category relation, the relation will be dropped with complete ripple down.

6. The entity is deleted from the model.

7. The primary name and all aliases for the entity will also be deleted.

8. All description text for the entity will be deleted.

9. Entities will not be dropped when mappings exist to records or fields or if the entity participates in unions, partitions, or algorithms.

Examples:

```
drop entity ORDER_INFO;
```

DROP FIELD

Syntax:

```
drop {columns }  FIELD_NAME ... of  {table  }  REC_NAME
     {fields  }                     {record }
     {elements}                     {segment}
     {items   }

   of  {database}  DB_NAME   ;
       {pcb     }
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   columns/fields/elements/items
   table/record/segment
   database/pcb.

2. This command deletes all the fields specified, and its associations, provided there are no associated mappings.

3. All description text will be deleted for the data field.

4. Subcomponent fields are also dropped.

5. All redefinitions of this field shall also be dropped and any of its subcomponents.

6. Dropping a redefinition field does not affect the original field.

7. Be aware that a dropped field may leave an incomplete record definition in the CDM.

Examples:

```
drop columns C1 C2 of table TAB_OR of database ORC_DB;

drop elements SI_ID SI_QTY of pcb IMS_DB ;
```

Syntax:

```
drop host  HOST_ID ... ;
```

Semantics:

1. HOST_ID must already be defined.
2. If the host is cross referenced with a DBMS, or any database is associated with the host, the host will not be dropped.
3. All description text for this host will also be deleted.

Examples:

```
drop host IBM HP;
```

DROP KEYWORD

Syntax:

drop keyword KEY_WORD ...;

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   keyword

2. The keyword reference is deleted from any entity class, attribute class and relation class.

3. The keyword itself is deleted.

4. Any description text associated with the keyword is also dropped.

5. NOTE: This command will drop a keyword irrespective of the model.

Examples:

drop keyword KEY_ADDRESS ;

DROP LUW

Syntax:

drop luw LUW_NAME;

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   Luw

2. The Logical unit of work is deleted.

3. All RP_Mains are deleted from the logical unit of work.

4. The logical unit of work will not be deleted if there are modules precompiled against it. Instead a list of the modules will be displayed, and then the Drop Module command can be used to drop them.

Examples:

drop luw REPORTS;

DROP MAP

Syntax:

- for attribute:

```
drop map EC_NAME.  {TAG_NAME for preference PREF_NUMB}
                   {*                                 };
```

- for link relation and category relation:

```
drop map EC_NAME1  RC_NAME  [EC_NAME2];
```

Semantics:

1. All mappings for a relation class must be dropped together.
2. When dropping an AUC mapping, if preference number 1 is specified, all mappings will be dropped since there must always be a primary mapping.
3. When EC_NAME.* is specified, all tags that are mapped for that entity for all preferences will be dropped.
4. The mapping will not be dropped if the tag is mapped thru a complex mapping. To delete this mapping, use DROP ALGORITHM.
5. To drop individual Relation Record Set mapping, use the ALTER MAP command.
6. When EC_NAME.* is specified, all entity to record mappings for the entity specified will be dropped along with the distributed rules for that entity.
7. Link relations require independent_name, RC_name and dependent_entity name. Category relations require generic_entity and rc_name.

Examples:

```
drop map PART.SIZE for preference 1;

drop map EMPLOYEE POSSESSES SKILL ;
```

Syntax:

```
drop model MODEL_NAME;
```

Semantics:

1. MODEL_NAME must exist.

2. Everything associated with a model will be dropped from the CDM. This is: all entities, owned attributes, attribute uses, keys, key members, inherited attribute uses, link and category relations and complete relations and all attributes. The descriptions, aliases and keywords for the entities, attributes and relations of the model will be dropped.

3. The MODEL_NAME itself and its descriptions will be deleted.

4. The INTEGRATED_MODEL cannot be dropped.

Examples:

```
DROP MODEL X;
```

Syntax:

```
drop module MOD_NAME ...;
```

Semantics:

1. The module itself will be dropped and all module parameters for each module to be dropped will also be dropped.

2. Any module still associated with a complex mapping algorithm cannot be dropped. Use Drop Algorithm to delete the algorithm.

3. A module generated by the NDML precompiler can also be dropped, provided it is not an RP-Main.

Examples:

```
drop module N51  N5510  OUTPART;
```

DROP PARTITION

Syntax:

drop partition [INTEGER1] of entity EC_NAME ;

Semantics:

1. The partitions of the entity named will be dropped.

2. This command will not affect the AUC mappings of the named entity in any way. The user is responsible for deleting any partition mappings of the specified entity, prior to using the DROP PARTITION command.

3. INTEGER1 unless specified will default to 1.

Examples:

drop partition of entity orders;

Syntax:

```
drop psb PSB_NAME ... ;
```

Semantics:

1. The PSB_NAME must not have any databases (pcb's) associated with it. To drop the PCB, use the DROP DATABASE command.

Examples:

```
drop psb ABCD;
```

DROP RECORD

Syntax:

```
drop   {table  }  REC_NAME  of  {database}  DB_NAME ;
       {record }                {pcb     }
       {segment}
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   table/record/segment
   database/pcb

2. This command deletes the record specified and all associated fields, areas, and sets provided there are no associated mappings.

3. All description text will be deleted for the record/table/segment and associated fields.

Examples:

```
drop table  TAB_OR  of database  ORC_DB ;

drop segment  SEG1  of pcb  IMS_DB ;
```

Syntax:

```
drop relation { EC_NAME_1 RC_NAME EC_NAME_2 } ...;
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model.:

   link relation, independent entity, dependent entity, model

2. If a key had been migrated, it is "unmigrated" i.e., the attribute use and inherited attributes are removed from the dependent entity and from any other entities they have migrated.

3. The link relation and complete relation are deleted from the model.

4. The keywords associated with the relation are also dropped.

5. All description texts for the relation will be deleted.

6. Relations will not be dropped, when link relation to set mappings are present.

Examples:

```
drop relation SALES_PERSON MANAGES ACCOUNT ;
```

Syntax:

```
drop  { set  } SET_NAME  of {database}  DB_NAME ;
      { path }              {pcb     }
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   set/path
   database/pcb

2. This command deletes the set specified.

3. If the set maps to a tag or relation class, the set will not be dropped.

4. All description texts for the set will be deleted.

Examples:

```
drop  set   TEST1  of  database  TOT_DB;

drop  path  TEST2  of  pcb  IMS_DB;
```

Syntax:

```
drop union of record DB_NAME.REC_NAME ;
```

Semantics:

1. The record union named must previously exist.
2. This will not affect the definition of the record, the entity or any of its attribute mappings.

Examples:

```
drop union of record DB1.REC4;
```

Syntax:

```
drop view VIEW_NAME ...;
```

Semantics:

1. A valid VIEW must exist in the Common Data Model.
2. All project data items associated with the view being dropped are deleted.
3. The surrogate entity (user view) to relation mappings associated with the view are deleted.
4. All data items associated with the view are deleted. First a check is made to verify that the data item does not participate in a conversion algorithm.
5. The view is deleted from the CDM.
6. All description texts for the view and data items belonging to the view will be deleted.

Examples:

```
drop view SUPPLIES MATERIAL_LIST ;
```

Syntax:

```
HALT [WITH ROLLBACK];
```

Semantics:

1. The NDDL session will be terminated.
2. If commit mode is automatic, the rollback option will have no effect.
3. If commit mode is manual, the rollback option will undo any changes made since the last commit point. If the rollback option is not specified, all changes made since the last commit point will be committed.
4. Note: use of the user interface "HALT" key, see page I-6, in the interactive mode is equivalent in meaning to using a HALT; command. The function key will not perform a halt with rollback.
5. Regardless of the commit mode, a Halt without the Rollback option will commit all changes.

MERGE MODEL

Syntax:

```
merge model MODEL_NAME_1 with MODEL_NAME_2
      [into MODEL_NAME_3]
      [except [description] [alias] [keyword]];
```

Semantics:

1. MODEL_NAME_1 and MODEL_NAME_2 must exist.

2. The NDDL commands necessary to physically merge MODEL_NAME_1 and MODEL_NAME_2 are generated on a file or screen. Use "SET OUTPUT" to establish the output mode - whether to the screen or to a file.

3. If MODEL_NAME_3 is entered, MODEL_NAME_3 must not exist. The result of the merge will create MODEL_NAME_3. All attributes, entities, link and category relations, keys, aliases, keywords and descriptions of MODEL_NAME_1 will be generated for MODEL_NAME_3, unless excepted. The output is the same as output from a COPY MODEL of MODEL_NAME_1 to MODEL_NAME_3.

4. If MODEL_NAME_3 is omitted, the result of the merge will alter MODEL_NAME_1.

5. For each entity in MODEL_NAME_2, if the entity does not exist in MODEL_NAME_1, the generated commands are the same as the output of a COPY ENTITY from MODEL_NAME_2 with relation.

6. For each entity in MODEL_NAME_2, if the entity does exist in MODEL_NAME_1, the generated commands are the same as the output of a COMBINE ENTITY from MODEL_NAME_2.

7. Refer to COMBINE ENTITY for a complete description of the EXCEPT clause.

Generated Commands

The generated NDDL commands should be examined for potential run-time errors.

1. A create/alter entity command may attempt to add an owned attribute to an entity when the attribute is already owned by another entity in the target model. The modeler must decide which entity should own the attribute and change the NDDL accordingly.

   NOTE: When an attribute is added to an entity and the attribute already exists in the target model, a comment is generated in the NDDL command following the attribute. The comment is:

   ```
   /* ATTRIBUTE MAY BE OWNED IN TARGET MODEL */
   ```

The comment will not cause an NDDL syntax error.

2. CREATE/ALTER CATEGORY commands should be checked for valid discriminating attributes and primary keys.

Examples:

```
merge model INTEGRATED_MODEL with MODEL_A;
merge model MODEL_A with MODEL_B into NEW_MODEL
    except alias keyword;
```

Syntax:

```
rename   {entity       EC_NAME1      to       EC_NAME2    ]
         {attribute    AC_NAME1      to       AC_NAME2    ]
         {keyword      KEYWORD1      to       KEYWORD2    ]
         {model        MODEL_NAME    to       MODEL_NAME2 ]
         {domain       DOM_NAME1     to       DOM_NAME2   ]
         {view         VIEW_NAME1    to       VIEW_NAME2  ]
         {dataitem     VIEW_NAME.DI_NAME1 to  DI_NAME2    ]
         {host         HOST_NAME1    to       HOST_NAME2  ]
         {relation     EC_NAME1 RC_NAME1      [EC_NAME2] to
                             RC_NAME2                      ]
         {database     DB_NAME1      to       DB_NAME2    ]
         {field        DB_NAME.REC_NAME.FIELD_NAME1 to
                       FIELD_NAME2                    ]
         {datatype     DATA_TYPE_NAME1        DATA_TYPE_NAME2]
         {set          DB_NAME.SET_NAME1 to   SET_NAME2   ]
         {record       DB_NAME.REC_NAME1 to   REC_NAME2   ]     ;
```

Semantics:

1. To complete the above command the following elements must exist in the Common Data Model:

   entity or attribute or keyword or model or database, or record or field or set or datatype or domain or view or relation or host or dataitem;

2. After determining that the above elements exist, and the elements to be renamed do not previously exist, the object is updated.

3. Rename relation command requires both the dependent entity name and the independent entity name for the link relation to be renamed.

4. Rename relation command requires generic entity name for the category relation to be renamed.

5. When renaming a keyword, and KEYWORD2 already exists, all references of KEYWORD1 are replaced with KEYWORD2, and KEYWORD1 is deleted.

6. Renaming the attribute will not affect the tag name in any entity if it is owned or inherited.

7. The current model must be established before renaming an entity, attribute or relation.

Examples:

```
rename entity INVOICE to ORDER;

rename relation INVOICE SUPPLIES ORDER_INFO to
                               DESCRIBES ;
```

Syntax:

```
ROLLBACK;
```

Semantics:

1. This command will roll back, or un-do, all changes made to the CDM since the last commit point.

2. If the current setting of commit mode is "automatic", this command will have no effect since any changes would have been committed upon the completion of the previous command or, if in error, rolled back.

Syntax:

```
SET COMMIT   {AUTOMATIC};
             { MANUAL  }
```

Semantics:

1. Upon activation of the NDDL processor, in batch or interactive mode, the commit feature will default to automatic. That is, CDM changes will be committed after every successful NDDL command and changes rolled back (undone) whenever a fatal or serious error is encountered.

2. The user may elect to control commit by specifying the "MANUAL" option. In this case, database changes will be committed only when the user issues the commit command. If manual is selected, the user may also issue the rollback command at any time to "undo" any changes that have been made since the last commit point.

3. In all cases, the user will be notified when CDM changes have been successfully committed or rolled back.

4. In interactive mode, the command entry form will display the commit option chosen, automatic or manual, as a reminder.

5. In all cases, user errors (fatal or serious) will cause a rollback to occur. The NDDL processor will not allow errors to corrupt the integrity of the CDM. Thus, if commit mode is manual, any wanted changes from a previous successful command, not yet committed by the user, will be lost.

SET OUTPUT

Syntax:

```
set output to ([new] file 'string');
              (screen              )
```

Semantics:

1. Generated NDDL commands will be directed to either a user specified file or to the screen.

2. When new is specified, any command that will write to a specified file will open a new version of that file. If the operating system does not support multiple versions of the same file name, the previous version will be deleted. If new is not specified, the contents are appended to the end of the file.

3. The default output for batch NDDL mode will be a previously named file.

4. The default output for interactive mode will be the screen.

5. The CRT form will indicate the current database, current model, current commit mode and current output device.

6. NDDL commands, if directed to the screen, may be edited and executed.

Examples:

```
set output to file 'dump it';
set output to screen;
```

SECTION 4

NDDL RESERVED WORDS

Following is a list of NDDL reserved words. These words cannot be used as a user-defined variable in any NDDL command.

ACCESSED
ACTIVE
ADD
AFTER
ALGORITHM
ALIAS
ALL
ALLOW
ALTER
AND
AREAS
AS
AT
ATTRIBUTE
AUTOMATIC
BACKUP
BEFORE
BETWEEN
BY
CHARACTER
CHECK
CLASS
COLUMN
COLUMNS
COMBINE
COMMA
COMMENT
COMMIT
COMPARE
CONSTANT
COPY
CREATE
DATA
DATABASE
DATAFIELD
DATAITEM
DATATYPE
DBMS
DECIMAL
DEFINE
DEPENDING
DESCRIBE
DESCRIPTION
DIRECTLY
DIRECTORY
DISALLOW
DISTINCT

```
DOMAIN
DOT
DROP
DUPLICATE
ELEMENT
ELEMENT
ENTITY
EQ
EXCEPT
EXIT
FEEDBACK
FIELD
FIELDS
FILE
FILE_NAME
FILES
FILLER
FLOAT
FOR
FROM
HALT
HIGH_VALUES
HOST
IN
INCLUDE
INDEXED
INHERITED
INTEGER
INTO
IS
ITEM
ITEMS
KEY
KEYWORD
KNOWN
LENGTH
LINKED
LOCATED
LOW_VALUES
LPAREN
MANUAL
MANY
MAP
MEMBER
MERGE
MIGRATES
MODEL
MODULE
NAMED
NEW
NILL
NONKEY
NOT
NTM
NULL
NUMERIC
OCCURS
OF
ON
```

OPTIONAL
ORIGINAL_SOURCE
OUTPUT
OWNED
OWNERSHIP
PACKED
PARAMETERS
PARTITION
PASSIVE
PASSWORD
PATH
PCB
POSITION
PREFERENCE
PRIMARY
PSB
QUOTE
RANGE
RECORD
REDEFINES
REDUNDANT
RELATING
RELATION
RELATIONAL
RENAME
REPLICATION
REQUIRED
RETRIEVAL
ROLLBACK
RPAREN
SCHEMA
SCREEN
SECOND
SEGMENT
SELECT
SET
SIGNED
SIZE
SPACES
STANDARD
START
STATUS
STORES
STRUCTURE
SUBSCHEMA
SUBSTITUTING
TABLE
THRU
TO
TYPE
UNION
UNIQUE
UNKNOWN
UNSIGNED
UPDATE
USERDATATYPE
USING
VALUE
VIEW

```
WHEN
WHERE
WITH
ZEROES
```

SECTION 5

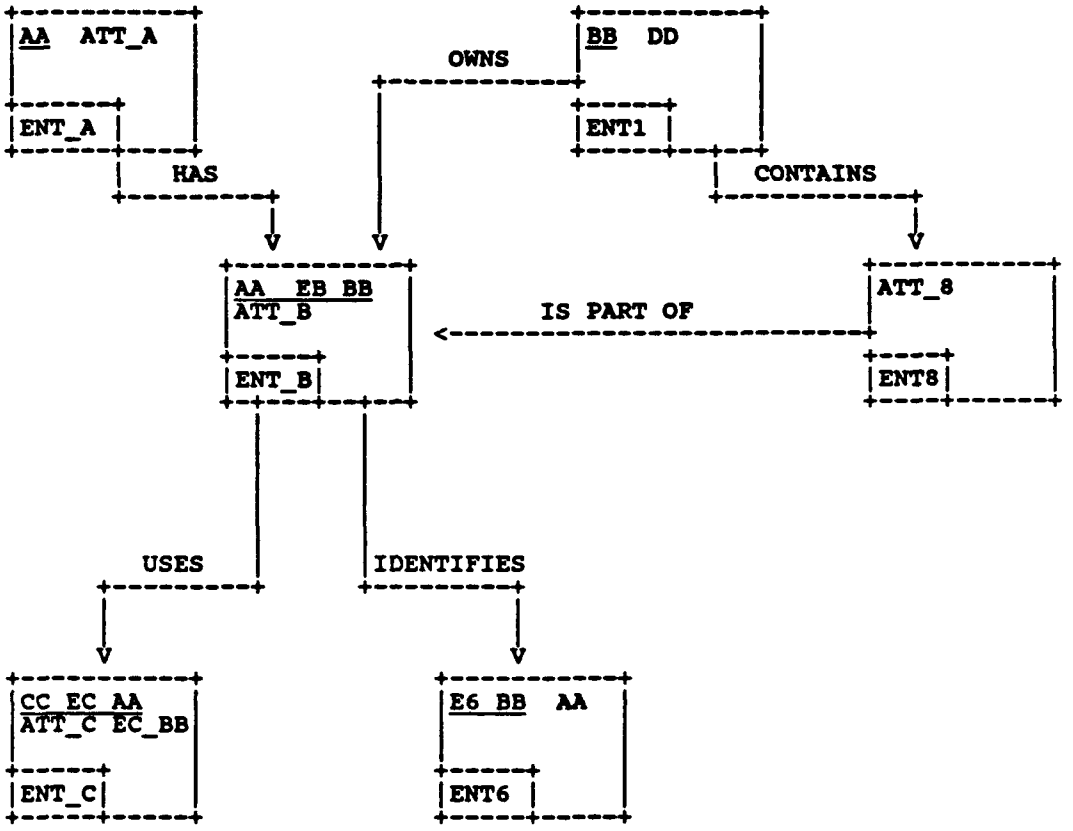
NDDL MODELING COMMANDS

This section will give an example of each modeling command and its expected results. The examples presented will use three models as a basis for all processing:

AUGIE_MOD
SALTY
CRUMMY_MOD

The following pages contain IDEF-1 diagrams for each model and the NDDL required to create each model.

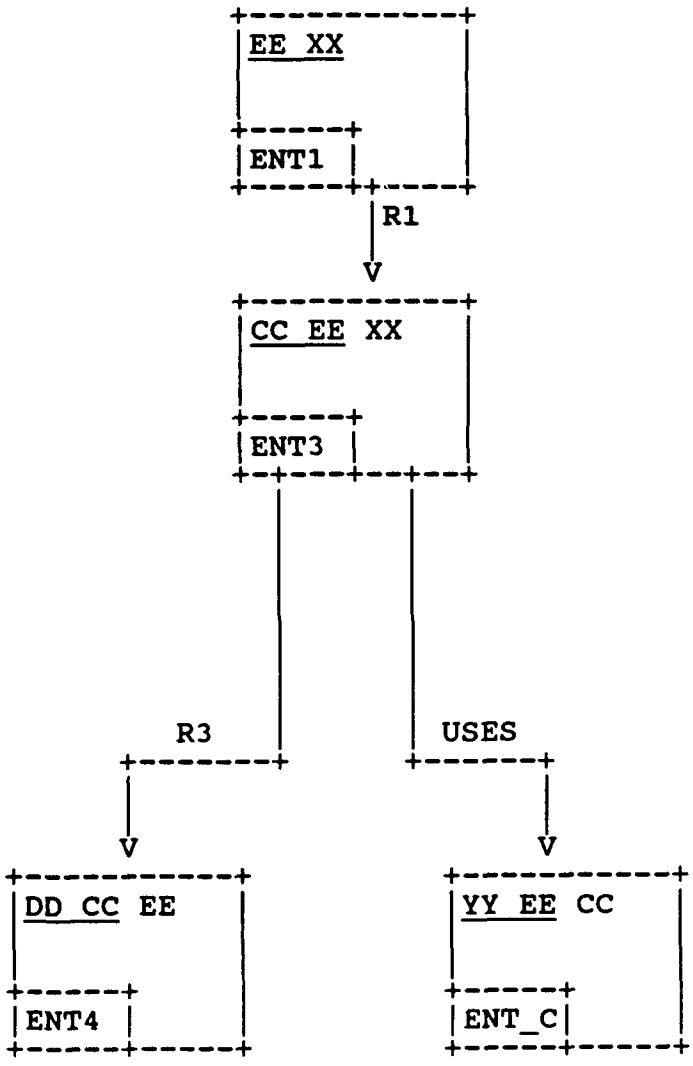
MODEL: AUGIE_MOD



NDDL for Model AUGIE_MOD

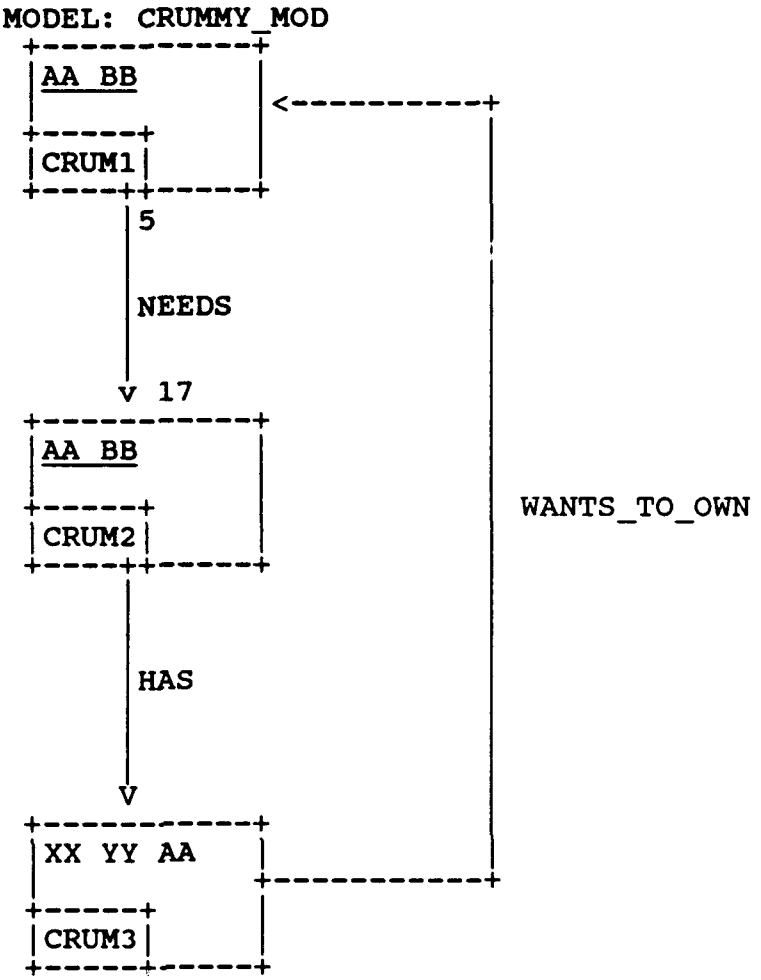
```
CREATE MODEL AUGIE_MOD;
CREATE ATTRIBUTE AA DOMAIN CHARACTER_NAME KEYWORD AA KEYWORD;
CREATE ATTRIBUTE ATT_A DOMAIN CHARACTER_NAME KEYWORD ATTA_BOY;
CREATE ATTRIBUTE BB;
CREATE ATTRIBUTE DD DOMAIN CHARACTER_NAME;
CREATE ATTRIBUTE ATT_8;
CREATE ATTRIBUTE CC DOMAIN CHARACTER_NAME KEYWORD CC_KEYWORD;
CREATE ATTRIBUTE ATT_B DOMAIN CHARACTER_NAME;
CREATE ATTRIBUTE ATT_C DOMAIN CHARACTER_NAME;
CREATE ALIAS ATTRIBUTE ATT_A IS ATTA_ALIAS;
CREATE ALIAS ATTRIBUTE BB IS BB_ALIAS;
CREATE ENTITY ENT_A KEY EAK = AA OWNED ATTRIBUTE AA ATT_A;
CREATE ENTITY ENT_B OWNED ATTRIBUTE ATT_B;
CREATE ENTITY ENT_C OWNED ATTRIBUTE CC ATT_C KEYWORD K3 K10;
CREATE ENTITY ENT1 KEY E1K = BB OWNED ATTRIBUTE BB DD;
CREATE ENTITY ENT8 OWNED ATTRIBUTE ATT_8;
CREATE ENTITY ENT6;
CREATE RELATION ENT_A HAS ENT_B MIGRATES EAK;
CREATE RELATION ENT1 OWNS ENT_B MIGRATES E1K SET EB_BB = BB;
ALTER ENTITY ENT_B ADD KEY EBK = AA EB_BB;
CREATE RELATION ENT1 CONTAINS ENT8;
CREATE RELATION ENT_B USES ENT_C MIGRATES EBK
     SET EC_AA = AA EC_BB = EB_BB
         KEYWORD USES_KW USES_KEYWORD;
ALTER ENTITY ENT_C ADD KEY ECK = EC_AA CC;
CREATE RELATION ENT_B IDENTIFIES ENT6 MIGRATES EBK
     SET E6_BB = EB_BB
ALTER ENTITY ENT6 ADD KEY E6K = E6_BB;
CREATE RELATION ENT8 IS_PART_OF ENT_B;
DESCRIBE DEFINITION OF ATTRIBUTE BB
  "BB IS AN ATTRIBUTE IN MODEL AUGIE_MOD";
DESCRIBE EXAMPLE OF ATTRIBUTE BB
  "A RONCO CODAMATIC IS AN EXAMPLE OF A BB ATTRIBUTE";
DESCRIBE DEFINITION OF ATTRIBUTE ATTA_ALIAS
  "THIS IS AN ALIAS OF ATT_A IN MODEL AUGIE_MOD";
DESCRIBE DEFINITION OF ATTRIBUTE ATT_C
  "ATT_C IS A PRIMARY ATTRIBUTE IN MODEL AUGIE_MOD";
DESCRIBE DEFINITION OF RELATION ENT_B USES ENT_C
  "ENT_B USES ENT_C IS A RELATION IN AUGIE_MOD";
DESCRIBE SOURCE OF RELATION ENT_B USES ENT_C
  "SOURCE OF RELATION IS A CRAZED ANALYST";
HALT;
```

MODEL: SALTY



NDDL for Model SALTY

```
CREATE MODEL SALTY;
CREATE ATTRIBUTE EE DOMAIN CHARACTER_NAME;
CREATE ATTRIBUTE CC DOMAIN CHARACTER_NAME;
CREATE ATTRIBUTE DD DOMAIN CHARACTER_NAME;
CREATE ATTRIBUTE XX DOMAIN CHARACTER_NAME;
CREATE ATTRIBUTE YY DOMAIN CHARACTER_NAME;
CREATE ENTITY ENT1 OWNED ATTRIBUTE EE XX KEYWORD K1 K2;
CREATE ENTITY ENT3 OWNED ATTRIBUTE CC KEYWORD K5;
CREATE ENTITY ENT4 OWNED ATTRIBUTE DD;
CREATE ENTITY ENT_C OWNED ATTRIBUTE YY KEYWORD USES_KW K3;
ALTER ENTITY ENT1 ADD KEY E1K = EE XX;
CREATE RELATION ENT1 R1 ENT3 MIGRATES E1K;
ALTER ENTITY ENT3 ADD KEY E3K3 = EE CC;
CREATE RELATION ENT3 R3 ENT4 MIGRATES E3K3;
ALTER ENTITY ENT4 ADD KEY E4K4 = CC DD;
CREATE RELATION ENT3 USES ENT_C MIGRATES E3K3 KEYWORD USES_KW;
ALTER ENTITY ENT_C ADD KEY OAK = EE YY;
HALT;
```


MODEL: CRUMMY_MOD
AA BB
CRUM1
5
NEEDS
v 17
AA BB
CRUM2
HAS
V
XX YY AA
CRUM3
WANTS_TO_OWN

NDDL for Model CRUMMY_MOD

```
CREATE MODEL CRUMMY_MOD;
CREATE ATTRIBUTE AA;
CREATE ATTRIBUTE BB DOMAIN CHARACTER_NAME;
CREATE ATTRIBUTE XX;
CREATE ATTRIBUTE YY;
CREATE ENTITY CRUM1 KEY C1K = AA BB;
CREATE ENTITY CRUM2;
CREATE ENTITY CRUM3 OWNED ATTRIBUTE XX YY;
CREATE RELATION 5 CRUM1 NEEDS 17 CRUM2 MIGRATES C1K;
ALTER ENTITY CRUM2 ADD KEY C2K1 = AA;
ALTER ENTITY CRUM2 ADD KEY C2K2 = AA BB;
CREATE RELATION CRUM2 HAS CRUM3 MIGRATES C2K1;
CREATE RELATION CRUM3 WANTS_TO_OWN CRUM1;
HALT;
```

CHECK MODEL

```
RESULT OF:
    SET OUTPUT TO FILE 'CRUMMY.OUT';
    CHECK MODEL CRUMMY_MOD;
```

GENERATED ERROR MESSAGES

```
MODEL BEING CHECKED:  CRUMMY_MOD 01-09-87  12:30:26
A NON SPECIFIC RELATION EXISTS BETWEEN CRUM1 NEEDS CRUM2
TAG AA FOR ENTITY CRUM1 HAS NO DOMAIN
TAG AA FOR ENTITY CRUM1 HAS NO DOMAIN
TAG AA FOR ENTITY CRUM1 HAS NO DOMAIN
C2K1 OF CRUM2 IS A DUPLICATE KEY OR SUBSET
ENTITY CRUM3 HAS NO ATTRIBUTE USE CLASS
ENTITY CRUM3 HAS NO KEY CLASS
TAG XX FOR ENTITY CRUM3 HAS NO DOMAIN
TAG YY FOR ENTITY CRUM3 HAS NO DOMAIN
MODEL CRUMMY_MOD HAS NO TOP ENTITY
MODEL CRUMMY_MOD HAS NO BOTTOM ENTITY
```

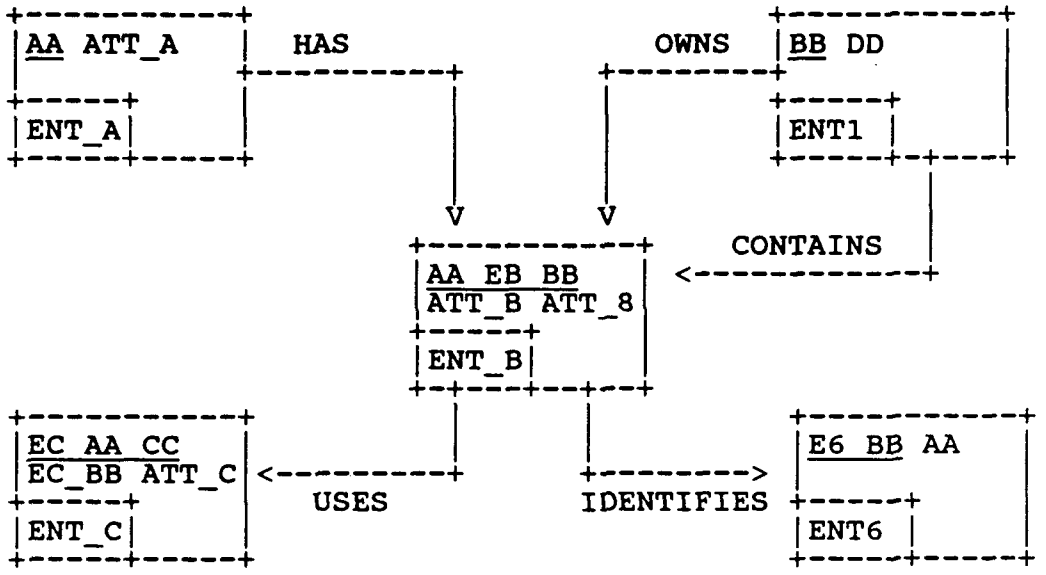
```
RESULT OF:
    CHECK MODEL SALTY;
```

GENERATED MESSAGE on file 'CRUMMY.OUT'

```
MODEL BEING CHECKED:  SALTY  01-09-87  1:25:07
MODEL SALTY HAS BEEN CHECKED
```

COMBINE ENTITY (Intra Model)

MODEL: AUGIE_MOD
RESULTS OF: COMBINE ENTITY ENT8 INTO ENT_B ;



```
ALTER MODEL AUGIE_MOD;
SET OUTPUT TO FILE 'CMBENT.ZZZ';
COMBINE ENTITY ENT8 INTO ENT_B;
```

GENERATED NDDL on file CMBENT.ZZZ

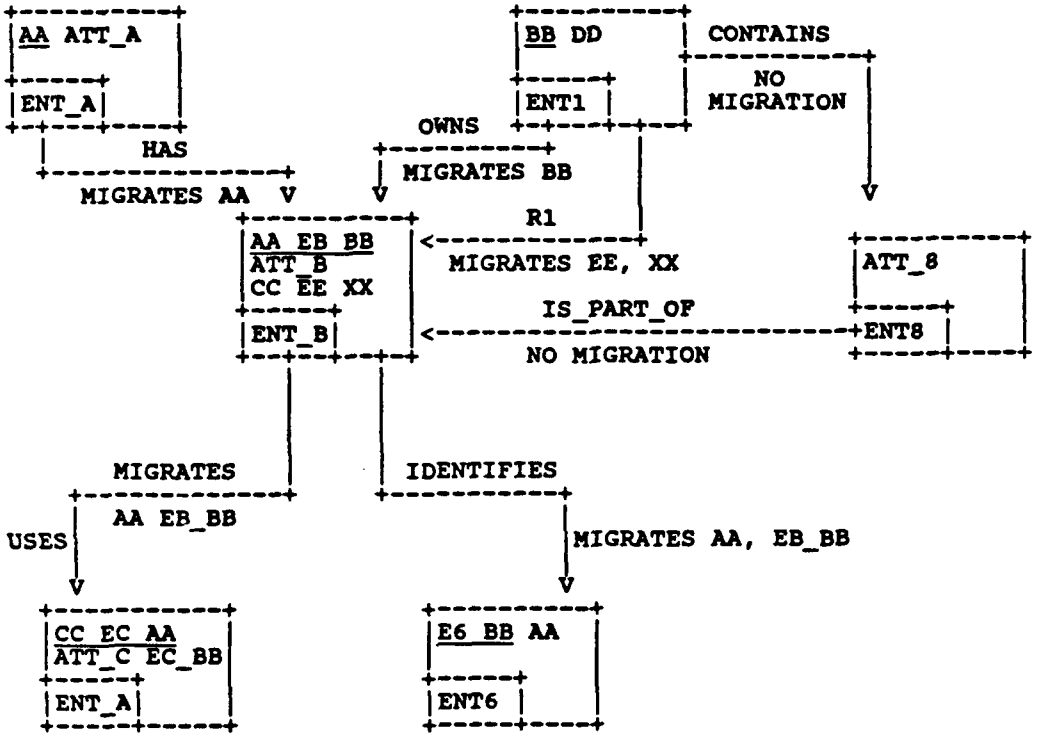
```
ALTER MODEL AUGIE_MOD;
DROP RELATION ENT8 IS_PART_OF ENT_B;
DROP ENTITY ENT8;
ALTER ENTITY ENT_B ADD OWNED ATTRIBUTE ATT_8
     /*THIS ATTRIBUTE MAY BE OWNED IN TARGET MODEL */;
CREATE ALIAS ENTITY ENT_B IS ENT8;
CREATE RELATION 1 ENT1 CONTAINS 0: MANY ENT_B;
```

SIGNIFICANT POINTS

1. DROPPING ENTITY ENT8 MAY HAVE SERIOUS CONSEQUENCES IF ITS KEY IS MIGRATED DOWN SEVERAL LEVELS.

COMBINE ENTITY (Inter Model)

MODEL: AUGIE_MOD
RESULTS OF: COMBINE ENTITY ENT3 FROM MODEL
SALTY INTO ENT_B;



```
ALTER MODEL AUGIE_MOD;
SET OUTPUT TO FILE 'CMBENT.ZZZ';
COMBINE ENTITY ENT3 FROM MODEL SALTY INTO ENT_B;
```

GENERATED NDDL on file CMBENT.ZZZ

```
ALTER MODEL AUGIE_MOD;
ALTER ENTITY ENT_B ADD OWNED ATTRIBUTE CC
    /* THIS ATTRIBUTE MAY BE OWNED IN TARGET MODEL */
     KEYWORD KS;
CREATE ALIAS ENTITY ENT_B IS ENT3;
CREATE RELATION 1 ENT1 R1 0: MANY ENT_B MIGRATES
     E1K SET EE = EE XX = XX;
ALTER ENTITY ENT_B ADD KEY E3K3 = CC EE;
```

SIGNIFICANT POINTS:

1. CC CANNOT BE OWNED BY ENT_B SINCE IT IS ALREADY OWNED BY ENT_C. CC MUST BE RENAMED.

2. EE AND XX CANNOT BE MIGRATED FROM ENT1 SINCE IT IS NEITHER OWNED NOR INHERITED BY ENT1.

3. THE RELATIION ENT3 R3 ENT4 IN MODEL SALTY IS NOT ADDED TO AUGIE_MOD SINCE ENT4 DOES NOT EXIST IN AUGIE_MOD.

4. RELATION ENT1 R1 ENT_B IS MIGRATING E1K (E1K = EE XX) OF MODEL SALTY, BUT THIS VERY SAME KEY ALREADY EXISTS IN MODEL AUGIE_MOD (E1K = BB)

5. RELATION ENT3 USES ENT_C IN MODEL SALTY ALREADY EXISTS AS ENT_B USES ENT_C IN MODEL AUGIE_MOD.

COMPARE MODEL

```
RESULT OF:
    SET OUTPUT TO FILE 'CMP.OUT';
    COMPARE MODEL AUGIE_MOD WITH SALTY;
```

GENERATED MESSAGES on file CMP.OUT

```
ENTITY OR ALIAS ENT_C EXISTS IN BOTH MODELS
ENTITY OR ALIAS ENT1 EXISTS IN BOTH MODELS
ATTRIBUTE OR ALIAS DD EXISTS IN BOTH MODELS
ATTRIBUTE OR ALIAS CC EXISTS IN BOTH MODELS
ENTITY ENT_C AND ENTITY ENT_C HAVE MATCHING KEYWORD K3
RELATION USES AND RELATION USES HAVE MATCHING KEYWORD USES_KW
```

COPY ATTRIBUTE

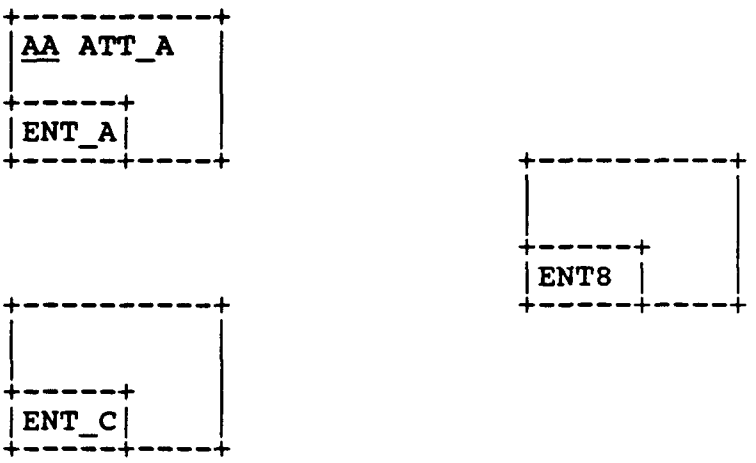
```
MODEL:  GENERAL PURPOSE
RESULT OF:  SET OUTPUT TO FILE 'COPATT.222';
            COPY ATTRIBUTE ATT_A FROM
            MODEL AUGIE_MOD TO
            NEWATT_A;
```

GENERATED NDDL ON FILE COPATT.ZZZ

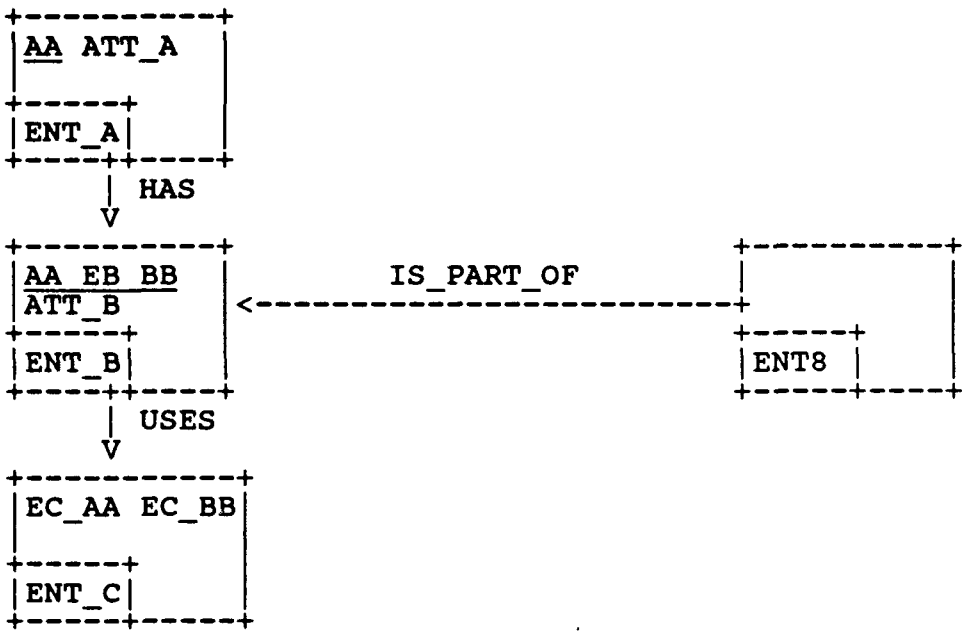
```
ALTER MODEL GENERAL PURPOSE;
CREATE ATTRIBUTE NEWATT_A DOMAIN CHARACTER_NAME
    KEYWORD ATTA_BOY:
DESCRIBE DEFINITION OF ATTRIBUTE
NEWATT_A
"THIS IS AN ALIAS OF ATT_A IN MODEL
AUGIE_MOD";
CREATE ALIAS ATTRIBUTE NEWATT_A IS
ATTA_ALIAS;
```

COPY ENTITY...WITH RELATION

MODEL: COPYREL



MODEL: COPYREL
RESULT OF: COPY ENTITY ENT_B FROM MODEL
AUGIE_MOD WITH RELATION;



```
ALTER MODEL COPYREL;
SET OUTPUT TO FILE 'iCOPENT.YYY';
COPY ENTITY ENT_B FROM MODEL AUGIE_MOD WITH RELATION;
```

GENERATED NDDL ON FILE COPENT.YYY

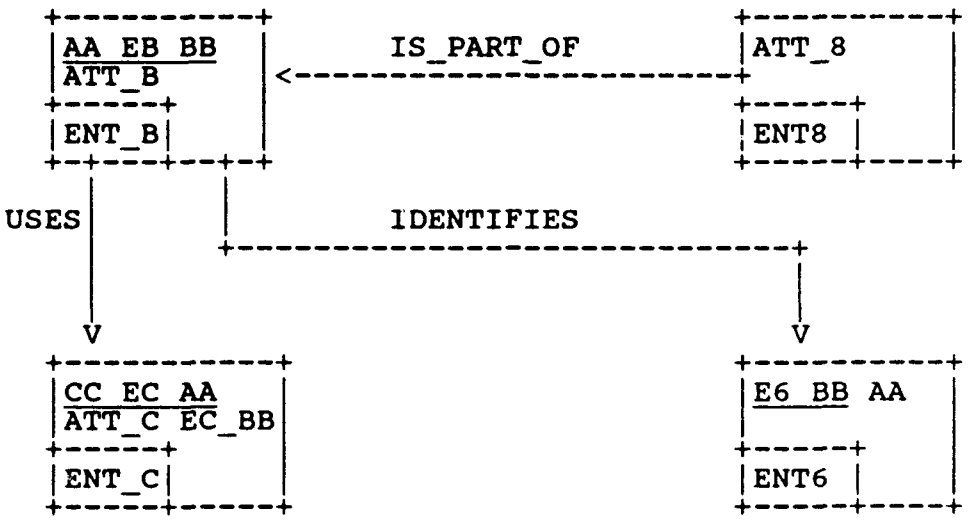
```
ALTER MODEL COYREL;
CREATE ATTRIBUTE ATT_B DOMAIN CHARACTER DOMAIN
    ;
CREATE ENTITY ENT_B OWNED ATTRIBUTE ATT_B;
CREATE RELATION 1 ENT_A HAS O: MANY ENT_B MIGRATES
    EAK SET AA = AA;
CREATE RELATION 1 ENT8 IS_PART_OF O: MANY ENT_B;
ALTER ENTITY ENT_B ADD KEY
    EBK = AA EB_BB;
CREATE RELATION 1 ENT_B USES O: MANY ENT_C MIGRATES
    EBK SET EC_AA = AA EC_BB = EB_BB
    KEYWORD USES_KW USES_KEYWORD;
DESCRIBE DEFINITION OF RELATION ENT_B USES ENT_C
"
ENT_B USES ENT_C IS A RELATION IN AUGIE_MOD
"
    ;
DESCRIBE SOURCE OF RELATION ENT_B USES ENT_C
"
SOURCE OF RELATION IS A CRAZED ANALYST
"
    ;
```

SIGNIFICANT POINTS:

1. ENT1 DOES NOT EXIST IN MODEL COPYREL HENCE THE RELATION ENT1 OWNS ENT_B IS NOT COPIED.

2. ATTRIBUTE EB_BB MUST BE CREATED PRIOR TO BEING INCLUDED AS PART OF THE KEY EBK.

MODEL: GENERAL-PURPOSE
RESULT OF: COPY ENTITY ENT8 FROM MODEL
AUGIE_MOD WITH STRUCTURE;



```
ALTER MODEL GENERALPURPOSE
SET OUTPUT TO FILE 'COPENT.ZZZ';
COPY ENTITY ENT8 FROM MODEL AUGIE_MOD WITH STRUCTURE;
```

GENERATED NDDL ON FILE COPENT.ZZZ

```
ALTER MODEL GENERALPURPOSE;
CREATE ATTRIBUTE ATT_8 DOMAIN UNDEFINED
    ;
CREATE ENTITY ENT8 OWNED ATTRIBUTE ATT_8;
CREATE ATTRIBUTE CC DOMAIN CHARACTER_NAME
    KEYWORD CC_KEYWORD;
CREATE ATTRIBUTE ATT_B DOMAIN CHARACTER_NAME
    ;
CREATE ATTRIBUTE ATT_C DOMAIN CHARACTER_NAME
    ;
DESCRIBE DEFINITION OF ATTRIBUTE IN MODEL AUGIE_MOD
"
ATT_C IS A PRIMARY ATTRIBUTE IN MODEL AUGIE_MOD
"
    ;
CREATE ENTITY ENT_B
    OWNED ATTRIBUTE ATT_B
    ;
CREATE ENTITY ENT_C
    OWNED ATTRIBUTE CC ATT_C
    KEYWORD K3 K10;
CREATE ENTITY ENT6
    ;
CREATE RELATION 1 ENT8 IS_KPART_OF O: MANY ENT_B
    ;
ALTER ENTITY ENT_B ADD KEY
    EBK = AA EB_BB;
CREATE RELATION 1 ENT_B USES O: MANY ENT_C
    MIGRATES EBK SET EC_AA = AA EC_BB = EB_BB
    KEYWORD USES_KW USES_KEYWORD;
```

```
DESCRIBE DEFINITION OF RELATION ENT_B USES ENT_C
"
ENT_B USES ENT_C IS A RELATION IN AUGIE_MOD
"
    ;
DESCRIBE SOURCE OF RELATION ENT_B USES ENT_C
"
SOURCE OF RELATION IS A CRAZED ANALYST
"
    ;
ALTER ENTITY ENT_C ADD KEY
    ECK = EC_AA CC;
CREATE RELATION 1 ENT_B IDENTIFIES O: MANY ENT6
    MIGRATES EBK SET AA = AA E6_BB = EB_BB
    ;
ALTER ENTITY ENT6 ADD KEY
    E6K = E6_BB;
```

SIGNIFICANT POINTS:

1. ATTRIBUTES AA AND EB_BB MUST BE CREATED PRIOR TO INCLUDING AS PART OF THE KEY EBK.

COPY MODEL

```
SET OUTPUT TO FILE 'COPMOD.XXX';
COPY MODEL FROM MODEL AUGIE_MOD TO AUGIE_COPY;
```

GENERATED NDDL ON FILE COPMOD.XXX

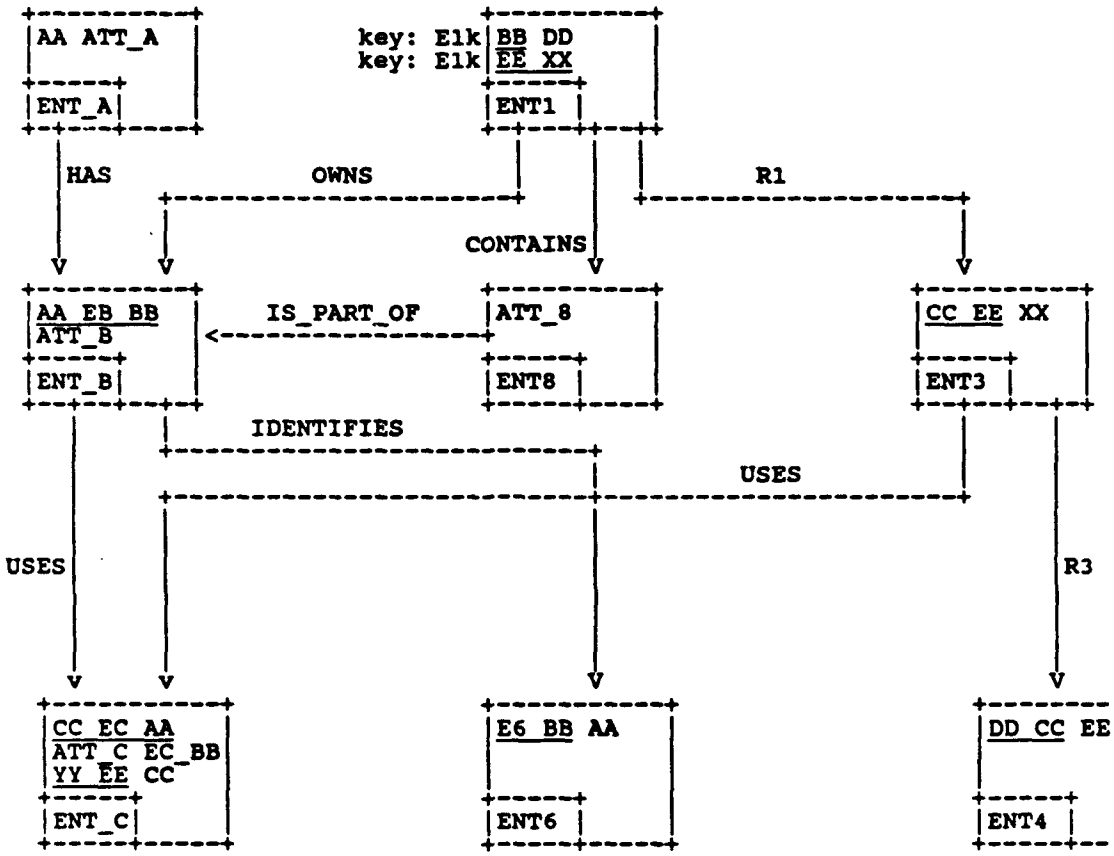
```
CREATE MODEL AUGIE_COPY;
CREATE ATTRIBUTE AA DOMAIN CHARACTER_NAME
    KEYWORD AA_KEYWORD;
CREATE ATTRIBUTE ATT_A DOMAIN CHARACTER_NAME
    KEYWORD ATTA_BOY;
DESCRIBE DEFINITION OF ATTRIBUTE ATT_A
"
THIS IS AN ALIAS OF ATT_A IN MODEL AUGIE_MOD
"
    ;
CREATE ALIAS ATTRIBUTE
    ATT_A IS ATTA_ALIAS;
CREATE ATTRIBUTE BB DOMAIN UNDEFINED
    ;
DESCRIBE DEFINITION OF ATTRIBUTE BB
"
BB IS AN ATTRIBUTE IN MODEL AUGIE_MOD
"
    ;
DESCRIBE EXAMPLE OF ATTRIBUTE BB
"
A RONCO CONDAMATIC IS AN EXAMPLE OF A BB ATTRIBUTE
"
    ;
CREATE ALIAS ATTRIBUTE
    BB IS BB_ALIAS;
CREATE ATTRIBUTE DD DOMAIN CHARACTER_NAME
    ;
CREATE ATTRIBUTE ATT_8 DOMAIN UNDEFINED
    ;
CREATE ATTRIBUTE CC DOMAIN CHARACTER_NAME
    KEYWORD CC_KEYWORD;
CREATE ATTRIBUTE ATT_B DOMAIN CHARACTER_NAME
    ;
CREATE ATTRIBUTE ATT_C DOMAIN CHARACTER_NAME
    ;
```

```
DESCRIBE DEFINITION OF ATTRIBUTE ATT_C
"
ATT_C IS A PRIMARY ATTRIBUTE IN MODEL AUGIE_MOD
"
    ;
CREATE ENTITY ENT_A
    OWNED ATTRIBUTE AA ATT_A
    ;
CREATE ENTITYT ENT_B
    OWNED ATTRIBUTE ATT_B
    ;
CREATE ENTITYT ENT_C
    OWNED ATTRIBUTE CC ATT_C
    KEKYWORD K3 K10;
CREATE ENTITY ENT1
    OWNED ATTRIBUTE BB DD
    ;
CREATE ENTITY ENT8
    OWNED ATTRIBUTE ATT_8
    ;
CREATE ENTITY ENT6
    ;
ALTER ENTITY ENT_A ADD KEY
    EAK = AA;
ALTER ENTITY ENT1 ADD KEY
    E1K = BB;
CREATE RELATION 1 ENT_A HAS O: MANY ENT_B MIGRATES EAK
    SET AA = AA
    ;
CREATE RELATION 1 ENT1 OWNS O: MANY ENT_B MIGRATES E1K
    SET EB_BB = BB
    ;
ALTER ENTITY ENT_B ADD KEY
    EBK = AA EB_BB;
CREATE RELATION 1 ENT1 CONTAINS O: MANY ENT8
CREATE RELATION 1 ENT8 IS_PART_OF O: MANY ENT_B;
    ;
CREATE RELATION 1 ENT_B USES O: MANY ENT_C MIGRATES EBK
    SET EC_AA = AA EC_BB = EB_BB
    KEYWORD USES_KW USES_KEYWORD;
DESCRIBE DEFINITION OF RELATION ENT_B USES ENT_C
"
ENT_B USES ENT_C IS A RELATION IN AUGIE_MOD
"
    ;
```

```
DESCRIBE SOURCE OF RELATION ENT_B USES ENT_C
"
SOURCE OF RELATION IS A CRAZED ANALYST
"
    ;
ALTER ENTITY ENT_CC ADD KEY
    ECK = EC_AA CC;
CREATE RELATION 1 ENT_B IDENTIFIES O: MANY ENT6
    MIGRATES EBK SET AA = AA E6_BB = EB_BB
    ;
ALTER ENTITY ENT6 ADD KEYK
    E6K = E6_BB;
```

MERGE MODEL

MODEL: AUGIE_MOD
RESULTS OF: MERGE MODEL AUGIE_MOD WITH SALTY
EXCEPT DESCRIPTION ALIAS KEYWORD;



```
SET OUTPUT TO FILE 'MRGMOD.FIL';
MERGE MODEL AUGIE_MOD WITH SALTY
    EXCEPT DESCRIPTION ALIAS KEYWORD;
```

GENERATED NDDL ON FILE MRGMOD.FIL

```
ALTER MODEL AUGIE_MOD ;
CREATE ATTRIBUTE EE DOMAIN CHARACTER_NAME ;
CREATE ATTRIBUTE XX DOMAIN CHARACTER_NAME ;
ALTER ENTITY ENT1 ADD OWNED ATTRIBUTE EE XX ;
ALTER ENTITY ENT1 ADD
    KEY E1K = EE XX ;
CREATE ENTITY ENT3 OWNED ATTRIBUTE CC
    /* THIS ATTRIBUTE MAY BE OWNED IN TARGET MODEL */ ;
CREATE RELATION 1 ENT1 R1 O: MANY ENT3
    MIGRATES E1K SET
    EE = EE XX =XX ;
ALTER ENTITY ENT3 ADD
    KEY ESK3 = EE CC ;
CREATE ENTITY ENT4 OWNED ATTRIBUTE DD
    /* THIS ATTRIBUTE MAY BE OWNED IN TARGET MODEL */ ;
CREATE RELATION 1 ENT3 R3  O: MANY ENT4
    MIGRATES E3K3 SET
    EE = EE CC = CC ;
ALTER ENTITY ENT4 ADD
    KEY E4K4 = CC DD ;
CREATE ATTRIBUTE YY DOMAIN CHARACTER_NAME ;
ALTER ENTITY ENT_C ADD OWNED ATTRIBUTE YY :
CREATE RELATION 1 ENT3 USES O: MANY ENT_C
    MIGRATES E3K3 SET
    EE = EE CC = CC
ALTER ENTITY ENT_C ADD
    KEY ECKC = EE YY ;
```

SIGNIFICANT POINTS:

1. ENTITY ENT1 HAS 2 IDENTICAL KEYS.

2. ATTRIBUTE CC IS OWNED BY ENTITIES ENT_C AND ENT3; ALSO ATTRIBUTE DD IS OWNED BY ENTITIES ENT1 AND ENT4.

SECTION 6

MAPPING PHASES

CREATE/DEFINE

Phase 1:
CREATE CONCEPTUAL SCHEMA (ENTITIES, ATTRIBUTES, TAGS, RELATIONS)
DEFINE INTERNAL SCHEMA (DATABASES, RECORDS, FIELDS, SETS)
(Note: Insure that the internal schema definitions match up with how the physical database has been created)

Phase 2:
CREATE VIEW
CREATE PARTITION
CREATE UNION
DEFINE MODULE

Phase 3:
CREATE MAP
DEFINE ALGORITHM

DROP

Phase 1:
DROP ALGORITHM
DROP MAP

Phase 2:
DROP UNION
DROP PARTITION
DROP VIEW
DROP MODULE

Phase 3:
DROP CONCEPTUAL SCHEMA (ENTITIES, ATTRIBUTES, TAGS, RELATIONS)
DROP INTERNAL SCHEMA (RECORDS, FIELDS, SETS, DATABASES)

SECTION 7

DISPLAY CDM CONTENTS

To display the entire contents of the CDM:

```
Set output to file 'dump.it';
Copy domain all;
Copy host all;
Copy dbms all;
Copy model INTEGRATED_MODEL;

Copy view all;
Copy module all;
Copy database all;
Copy map for entity all;
Copy map for relation all;
```

This is probably best done in batch.

SECTION 8

TABLE OF DATATYPES

Tables A-1 thru A-6 summarize the VAX-11 data type supported by VAX-11 C, COBOL, FORTRAN and NDML. If an entity contains a NO, the VAX-11 data type is not supported by that language.

TABLE A-1: SIGNED Integer Data Types

	SIGNED BYTE	SIGNED WORD	SIGNED LONGWORD	SIGNED QUADWORD
C	CHAR	SHORT INT	INT	NO
COBOL	NO	PIC S9(1-4) COMP	PIC S9(5-9) COMP	PIC S9 (10-18) COM
FORTRAN	BYTE	INTEGER*2	INTEGER*4 or INTEGER	NO
NDML	NO	I 1-4	I 5-9	I 10-18

TABLE A-2: UNSIGNED Integer Data Types

	UNSIGNED BYTE	UNSIGNED WORD	UNSIGNED LONGWORD	UNSIGNED QUADWORD
C	UNSIGNED CHAR	UNSIGNED SHORT INT	UNSIGNED INT	NO
COBOL	NO	PIC 9(1-4) COMP	PIC 9(5-9) COMP	PIC 9 (10-18) CO
FORTRAN	NO	NO	NO	NO
NDML	NO	NO	NO	NO

TABLE A-3: FLOATING Data Types

	FLOATING	D_FLOATING	G_FLOATING	H_FLOATING
C	FLOAT	DOUBLE	NO	NO
COBOL	(no PIC) COMP-1	no PIC) COMP-2	NO	NO
FORTRAN	REAL or REAL*4	REAL*8 +	REAL*8 +	REAL*16
NDML	F	NO	NO	NO

+ = A compiler switch may be necessary; this data type may not be compatible with another data type if used in the same program. See the language reference manual for this language for more information.

TABLE A-4: FLOATING COMPLEX Data Types

	FLOATING COMPLEX	D_FLOATING COMPLEX	G_FLOATING COMPLEX	H_FLOATING COMPLEX
C	NO	NO	NO	NO
COBOL	NO	NO	NO	NO
FORTRAN	COMPLEX or COMPLEX*8	COMPLEX*16 +	COMPLEX*16 +	COMLEX*32
NDML	NO	NO	NO	NO

+ = A compiler switch may be necessary; this data type may not be compatible with another data type if used in the same program. See the language reference manual for this language for more information.

TABLE A-5: CHARACTER Data Types

	CHARACTER
C	CHAR(n) +
COBOL	PIC X(n) +
FORTRAN	CHARACTER*n ++
NDML	C

+ = n is the length in bytes.

++ = n is less than or equal to 32,767.

TABLE A-6: NUMERIC Data Types

	UNSIGNED NUMERIC	LEFT SEPARATE NUMERIC	RIGHT SEPARATE NUMERIC
C	NO	NO	NO
COBOL	PIC 9(1-18) +	PIC S9(1-18) + SIGN LEADING SEPARATE	PIC S9(1-18) + SIGN TRAILING SEPARATE
FORTRAN	NO	NO	NO
NDML	N(1-18)	NO	NO

+ = There is a language restriction: the number of digits cannot be greater than 18.

TABLE A-6: NUMERIC Data Types (Cont.)

	LEFT OVERPUNCHED NUMERIC	RIGHT OVERPUNCHED NUMERIC	ZONED NUMERIC	PACKED DECIMA
C	NO	NO	NO	NO
COBOL	PIC S9(1-18) + SIGN LEADING	PIC S9(1-18) + SIGN TRAILING	NO	PIC S9(1- COMP-3
FORTRAN	NO	NO	NO	NO
NDML	NO	S (1-18)	NO	P (1-18)

+ = There is a language restriction: the number of digits cannot be greater than 18.

APENDIX A

NDDL ERROR LISTING

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
--COMMAND NOT COMMITTED--	PROCMD	WARNING	
--COMMAND SUCCESSFUL, COMMIT PERFORMED--	PROCMD	WARNING	
--COMMAND SUCCESSFUL--	PROCMD	WARNING	
--COMMAND SYNTAX CORRECT--	PROCMD	WARNING	
--COMMANDS ROLLED BACK--	PROCMD	WARNING	
--COMMIT PERFORMED--	PROCMD	WARNING	
<AC-NAME> KEYWORD NOT DELETED	DRPATT	ERROR	KEYWORD
<AREA-ID> AND <RT-ID> ASSOCIATION COULD NOT BE INSERTED	ALTREC	ERROR	RECORD
<AREA-NAME> COULD NOT BE INSERTED INTO DATA_BASE AREA	DEFREC	ERROR	DATABASE
<AUC-NAME> CAN NOT BE ADDED AS KEY MEMBER	ADDKC	ERROR	KEY
<AUC-NAME> IS AN INVALID KEY CLASS MEMBER	ADDKM	WARNING	ATTRIBUTE
<AUC-NAME> IS NOT IN KEY CLASS	MKRNLST	ERROR	RELATION
<DATA-TYPE-NAME> IS CURRENTLY USED BY THE VERIFICATION MODULE	DRPDT	ERROR	USER-DATATYPE
<DATA-TYPE-NAME> IS NOT DEFINED IN USER_DEF_DATA_TYPE	DEFFLDS	ERROR	RECORD
<DATA-TYPE-NAME> IS NOT DEFINED IN USER_DEF_DATATYPE	ADDFLDS	ERROR	RECORD
<DATA-TYPE-NAME> IS NOT DEFINED IN USER_DEF_DATA_TYPE	ALTFLDS	ERROR	DATAFIELD

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
<DATA-TYPE-NAME> IS REFERENCED IN DATA_ITEM OR <DATA-TYPE-NAME> IS REFERENCED IN DATA_FIELD	DRPDT	ERROR	USER-DATATYPE
<DBMS-NAME> IS NOT A LEGAL DBMS	DEFSET	ERROR	SET
<DF-ID> DOES NOT EXIST.	ALTFLDS	ERROR	DATAFIELD
<DF-ID> IS ALREADY DEFINED IN <RT-ID>	ADDFLDS	ERROR	RECORD
<DF-ID> SHOULD BE DEFINED BEFORE FIELD IT IS INDEXING	DEFFLDS	ERROR	RECORD
<DF-ID> WAS NOT PREVIOUSLY INDEXED DATAFIELD	ALTFLDS	WARNING	DATAFIELD
<DF-ID> WAS PREVIOUSLY INDEXED	ALTFLDS	ERROR	DATAFIELD
<EC-NAME> NOT FOUND ON ENTITY_NAME	ADDUNIN	ERROR	CSIS
<EC-NAME> NOT FOUND ON ENTITY_NAME	ALTUNIN	ERROR	CSIS
<ERR-CNTR-X> ERRORS FOUND WHEN VERIFYING VALUES ON VALUE TABLE	ALTVLRG	ERROR	DOMAIN
<ERR-CNTR-X> ERRORS WHEN VERIFYING RANGES ON DOMAIN_RANGE TABLE	ALTVLRG	ERROR	DOMAIN
<FDL-FILE-NAME(FDL-INDEX)> <FDL-HOST-ID(FDL-INDEX <FDL-MODULE-NAME(FDL-INDEX)>	DRPSMOD	WARNING	MODULE
<FIELD-NAME-1> DOES NOT EXIST	ADDFLDS	ERROR	RECORD
<FIELD-NAME-1> IS ALREADY INDEXING ANOTHER FIELD	ALTFLDS	ERROR	DATAFIELD
<HEADER-LINE>	DRPSMOD	WARNING	MODULE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
<INDEX-DFID(INDEX-IND)> SHOULD BE DEFINED BEFORE FIELD IT IS INDEXING	ADDFLDS	ERROR	RECORD
<MOD-ID> NOT DEFINED FOR PREFERENCE	DRPALG	ERROR	CSIS/CSES
<MOD-NAME> MODEL CANNOT BE COPIED	CPYMOD	ERROR	MODEL
<MOD-NAME> MODEL STRUCTURE IS INVALID	CPYMOD	ERROR	MODEL
<OBJ-TYPE> IS AN UNRECOGNIZED OBJ TYPE	VEROBJ	ERROR	
<OBJ-TYPE> IS AN UNRECONGNIZED OBJ TYPE	VEROBJ1	ERROR	OBJECT
<RC-NAME-LST> DOES NOT EXIST	AOAMGDP	ERROR	RELATION
<RT-ID>: NOT DELETED FROM DATA_FIELD	DRPREC	ERROR	RECORD
<RT-ID>: NOT DELETED FROM DB_AREA_ASSIGNMENT	DRPREC	ERROR	RECORD
<RT-ID>: NOT DELETED FROM LINKAGE_DATA_FIELD	DRPREC	ERROR	RECORD
<RT-ID>: NOT DELETED FROM RECORD-TYPE	DRPREC	ERROR	RECORD
<RT-ID>: TEXT DESC WAS NOT DELETED	DRPREC	ERROR	RECORD
<SET NAME> HAS ALREADY BEEN RC MAPPED	MAPRC	ERROR	SET
<SET-ID> ALREADY EXISTS	DEFSET	ERROR	SET
<SET-ID> ALREADY EXISTS	DEFSET	ERROR	SET

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
<TAG-NAME> IS NOT IN ENTITY <EC-NAME> DOES NOT HAVE SPECIFIED TAG NAME	BLVWLST	ERROR	VIEW
<TDFT-DFID(TDFT-INDEX)> IS A DEPENDING ON FIELD AND CANNOT BE A GROUP OR REPEATING OR INDEX.	FLDSEMS	ERROR	RECORD
<TDFT-DFID(TDFT-INDEX)> IS A DEPENDING ON FIELD AND MUST DEFINED AS NUMERIC DATA TYPE	FLDSEMS	ERROR	RECORD
<TDFT-DFID(TDFT-INDEX)> IS A REDEFINING FIELD AND CANNOT BE REPEATING OR AN INDEX	FLDSEMS	ERROR	RECORD
<TDFT-DFID(TDFT-INDEX)> IS AN INDEX FIELD WHICH CANNOT REDEFINE OR BE A GROUP OR A KEY OR A REPEATING FIELD	FLDSEMS	ERROR	RECORD
<TDFT-DFID(TDFT-INDEX)> MUST BE SAME TYPE KEY AS <TDFT-DFID(TEMP-INDEX)>	FLDSEMS	ERROR	RECORD
<attribute class name> ATTRIBUTE CLASS NOT DELETED	DRPATT	ERROR	ATTRIBUTE
<attribute class name> DESCRIPTION TEXT NOT DELETED	DRPATT	ERROR	ATTRIBUTE
<attribute class name> DOES NOT EXIST	DRPATT	ERROR	ATTRIBUTE
<attribute class name> DOES NOT EXIST	ALTATT	ERROR	ATTRIBUTE
<attribute class name> NAME NOT DELETED	DRPATT	ERROR	ATTRIBUTE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
<attribute class name> OWNED ATTRIBUTE NOT DELETED	DRPATT	ERROR	ATTRIBUTE
<member record name> DOES NOT EXIST FOR DB AND SET	FND1MEM	ERROR	SET
<set name> DOES NOT EXIST	FND1MEM	ERROR	SET
<set name> HAS ALREADY BEEN AUC MAPPED	MAPRC	ERROR	SET
<set name> MUST HAVE ONE MEMBER REC TYPE	FND1MEM	ERROR	SET
A CS-IS MAPPING ALREADY EXISTS FOR THIS TAG	ADDMAP	ERROR	CSIS
A CURRENT MODEL HAS NOT BEEN ESTABLISHED FOR THE SESSION	VEROBJ	ERROR	MODEL
A CURRENT MODEL HAS NOT BEEN ESTABLISHED FOR THE SESSION	VEROBJ1	ERROR	MODEL
A DATABASE ERROR HAS OCCURRED - DESCRIBE NOT PERFORMED	RDDESC	ERROR	DATABASE
A DATABASE ERROR HAS OCCURRED - DESCRIBE NOT PERFORMED	DDESC	ERROR	DATABASE
A DATABASE ERROR HAS OCURRED - MAPPING NOT PERFORMED	MAPRC	ERROR	CSIS/CSES
A DESCRIPTION EXISTS FOR OBJECT COPIED TO	SELDESC	ERROR	DESCRIPTION
A DESCRIPTION EXISTS FOR OBJECT COPIED TO	SEL1DSC	ERROR	DESCRIPTION
A MAPPING ALREADY EXISTS FOR THIS PREFERENCE	ADDMAP	ERROR	CSIS
A STANDARD DATA TYPE IS ALREADY DEFINED <DATA-TYPE-NAME>	ADDSTD	ERROR	USERDATATYPE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
A UNIQUE NUMBER CAN NOT BE ASSIGNED TO ATTRIBUTE	BLDATT	ERROR	ATTRIBUTE
A UNIQUE NUMBER CAN NOT BE ASSIGNED TO ATTRIBUTE <AC-NAME>	BLDATT1	ERROR	ATTRIBUTE
A UNIQUE NUMBER CAN NOT BE ASSIGNED TO MODEL <model-name> BEING CREATED	CRTMOD	ERROR	MODEL
ABBR-NAME MUST BE UNIQUE FOR EACH ENTITY	BLVWLST	ERROR	VIEW
ABBREVIATIONS MUST BE USED IF MORE THAN 1 ENTITY SPECIFIED	BLVWLST	ERROR	VIEW
AC KEYWORD ALREADY EXISTS	ADDKWA	ERROR	KEYWORD
ACTIVE MAPPINGS MUST HAVE PREF NUMBERS BETWEEN 1 AND 50	CRTMAP	ERROR	CSIS
ACTIVE MAPPINGS MUST HAVE PREF NUMBERS BETWEEN 1 AND 50	ALTMAP	ERROR	CSIS
ADD OR DROP MUST ALWAYS BE SPECIFIED	ALTREC	ERROR	RECORD
ADD/DROP CLAUSE CANNOT BE USED IF PREFERENCE NUMBERS WERE SWITCHED	ALTMAP	ERROR	CSIS
ADDFRM ERROR IN INITSES - RCODE IS <RCODE>	INITSES	ERROR	
ALGORITHM USING <MOD-ID> TO BE DELETED IS NOT DEFINED	DRPALG	ERROR	CSIS/CSES
ALGORITHM USING MODULE <MOD-ID> COULD NOT BE DELETED	DRPALG	ERROR	CSIS/CSES

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ALIAS <AC-NAME> DOES NOT EXIST <alias-id>	DRPALI	ERROR	ATTRIBUTE
ALIAS <ALIAS-ID> DOES NOT EXIST FOR ATTRIBUTE	ALTALI	ERROR	ATTRIBUTE
ALIAS <ALIAS-ID> DOES NOT EXIST FOR ENTITY	ALTALI	ERROR	ATTRIBUTE
ALIAS <alias-id> ALREADY EXISTS FOR ATTRIBUTE	CRTALI	ERROR	ATTRIBUTE
ALIAS <alias-id> ALREADY EXISTS FOR ENTITY	CRTALI	ERROR	ENTITY
ALIAS <ec-name> DOES NOT EXIST	DRPALI	ERROR	ENTITY
ALIAS COULD NOT BE CREATED FOR ATTRIBUTE <ob-id>	CRTALI	ERROR	ATTRIBUTE
ALIAS COULD NOT BE CREATED FOR ENTITY <obj-id>	CRTALI	ERROR	ENTITY
ALL ENTITY NUMBERS ASSIGNED - <EC-NAME> NOT PROCESSED	CRTENT	ERROR	ENTITY
ALL FIELDS MUST BELONG TO SAME RECORD	ALGCHK	ERROR	CSIS
ALL KEY CLASS NUMBERS ASSIGNED	ADDKC	ERROR	KEY
ALL KEYWORD NUMBERS ASSIGNED	ADDKW	ERROR	KEYWORD
ALL OTHER COMMANDS WILL BE SYNTAX CHECKED ONLY ---NO CHANGE MADE---	PROCMD	WARNING	
ALL RECORD TO ENTITY MAPS WILL BE DROPPED FOR THIS ENTITY SHOULD BE USED	DRPSMAP DROP	ERROR MAP	CSIS COMMAND

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ALL RELATION CLASS NUMBERS ASSIGNED	CRTREL	ERROR	RELATION
ALL TAGS MUST BELONG TO THE SAME ENTITY	ALGCHK	ERROR	CSIS
ALL VALUES AND RANGES DROPPED VERIF. MODULE DELETED= <RETV-APNAME>	ALTVMOD	WARNING	
ALTER ADD 'TO FIELD' INVALID UNLESS ENTITY IS HP	ADDMAP	ERROR	CSIS
ALTER COMMAND HAS INADVERTENTLY DELETED ALL PARTITIONS	ALTPART	ERROR	ENTITY
ALTER DROP 'FIELD' COMMAND WOULD LEAVE NO FIELDS IN MAP - USE DROP MAP COMMAND	DRPSMAP	ERROR	CSIS
ALTER DROP INVALID UNLESS ENTITY IS HZ_PART	DRPSMAP	ERROR	CSIS
ALTER FUNCTION INVALID FOR RC MAPPING	ALTMAP	ERROR	RELATION
ALTERING PREF NUM DOES NOT ALLOW CLASS/CATEGORY TO BE CHANGED	ALTMAP	ERROR	CSIS
ALTERING PREF NUM WOULD LEAVE TAG W/O A FIRST PREFERENCE	ALTMAP	ERROR	CSIS
ALTERING PREFERENCE WOULD LEAVE TAG WITHOUT A FIRST PREFERENCE	ALTMAP	ERROR	CSIS
AN ELEMENTARY NON-REPEATING FIELD MUST HAVE A DATA TYPE - SEE <TDFT-DFID(TDFT-INDEX)>	FLDSEMS	ERROR	RECORD
AN ENTITY IN MODEL <model-name> HAS MORE THAN 5 KEY CLASSES	ADDRCEC	ERROR	MODEL

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
AN UNEQUAL NUMBER OF ATTRIBUTES FOR SUBSTITUTION CANNOT BE PROCESSED	KCMSUBS	ERROR	ATTRIBUTE
AND RECORD <RT-ID> DO NOT EXIST ON CDM	ALTPART	ERROR	RECORD
AND RECORD TYPE <RT-ID> DO NOT EXIST	CRTPART	ERROR	RECORD
APPLICATION PRECOMPILED AGAINST DATBASE	CHKRPS	ERROR	DATABASE
APPLICATIONS EXIST AGAINST DATABASE	VERRCDB	ERROR	DATABASE
APPLICATIONS EXIST AGAINST DATABASE	VERRPDB	ERROR	DATABASE
AREA <AREA-ID> COULD NOT BE INSERTED	DEFAREA	ERROR	DATABASE
AREA <AREA-ID> COULD NOT BE INSERTED INTO DB_AREA_ASSIGNMENT	DEFREC	ERROR	DATABASE
AREA <AREA-ID> DOES NOT EXIST	DEFAREA	ERROR	DATABASE
AREA <AREA-ID> DOES NOT EXIST	ALTREC	ERROR	RECORD
AREA <AREA-NAME> ALREADY EXISTS	DEFCODL	WARNING	DATABASE
AREA <AREA-NAME> ALREADY EXISTS	ALTCODL	ERROR	DATABASE
AREA <AREA-NAME> DOES NOT EXIST	ALTCODL	ERROR	DATABASE
AREA ALREADY EXISTS IN DATA_ BASE_AREA	DEFREC	ERROR	DATABASE
AREA INFORMATION MISSING OR INCORRECT FOR CODASYL DBMS	DEFREC	ERROR	DATABASE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
AT LEAST 1 AREA/RECORD ASSOCIATION MUST EXIST FOR CODASYL DATA BASES	ALTREC	ERROR	RECORD
AT LEAST 2 RECORD PARTITIONS MUST EXIST FOR AN ENTITY	CRTPART	ERROR	ENTITY
ATTEMPT TO DELETE LAST AREA ASSOCIATED WITH DATA BASE	ALTCODL	ERROR	DATABASE
ATTEMPT TO DELETE LAST FILE ASSOCIATED WITH DATA BASE	ALTTOT	ERROR	DATABASE
ATTRIBUTE <AC-NAME2> ALREADY EXISTS	FCOPATT	ERROR	ATTRIBUTE
ATTRIBUTE <AC-NAME> ALREADY EXISTS IN THE MODEL	BLDATT1	WARNING	ATTRIBUTE
ATTRIBUTE <AC-NAME> CAN NOT BE CREATED	BLDATT1	ERROR	ATTRIBUTE
ATTRIBUTE <AC-NAME> COULD NOT BE DELETED	GETACAL	ERROR	ATTRIBUTE
ATTRIBUTE <AC-NAME> DOES NOT EXIST	ALTALI	ERROR	ATTRIBUTE
ATTRIBUTE <AC-NAME> IS ALREADY ↄWNED	ADDOAC	ERROR	ATTRIBUTE
ATTRIBUTE <AC-NAME> NOT INSERTED INOT ATTRIBUTE_NAME	ICOPATT	ERROR	ATTRIBUTE
ATTRIBUTE <AC-NAME> TO BE COPIED DOES NOT EXIST	COPATT	ERROR	ATTRIBUTE
ATTRIBUTE <ALIAS-ID> SYSTEM ERROR: UNABLE TO ALTER BIAS	ALTALI	ERROR	ATTRIBUTE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ATTRIBUTE <OBJ-ID-1> DOES NOT EXIST	VEROBJ	ERROR	ATTRIBUTE
ATTRIBUTE <OBJ-ID> DOES NOT EXIST	ALTALI	ERROR	ATTRIBUTE
ATTRIBUTE <OBJ-ID> SYSTEM ERROR: UNABLE TO ALTER BIAS	ALTALI	ERROR	ATTRIBUTE
ATTRIBUTE <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	ATTRIBUTE
ATTRIBUTE <TAG-NAME> DOES NOT EXIST IN THIS ENTITY	KCMSUSB	ERROR	ATTRIBUTE
ATTRIBUTE <TAG-NAME> DOES NOT EXITS IN THE ENTITY	KCMDRP	ERROR	ATTRIBUTE
ATTRIBUTE <TAG-NAME> HAS DOMAIN UNDEFINDED	CHKDOMS	ERROR	DOMAIN
ATTRIBUTE <TAG-NAME> HAS DOMAIN UNDEFINED	SMVIEW	ERROR	ATTRIBUTE
ATTRIBUTE <TAG-NAME> IS ALREADY A MEMBER OF THE SPECIFIED KEY	KCMSUBS	ERROR	ATTRIBUTE
ATTRIBUTE <TAG-NAME> IS NOT A MEMBER OF THE SPECIFIED KEY	KCMSUBS	ERROR	ATTRIBUTE
ATTRIBUTE <TAG-NAME> IS NOT ASSOCIATED WITH THIS ENTITY	KCMSUBS	ERROR	ATTRIBUTE
ATTRIBUTE <TAG-NAME> OR DOMAIN INFORMATION MISSING	MAPADF	ERROR	ATTRIBUTE
ATTRIBUTE <ac-name> ALREADY EXISTS IN THE MODEL	BLDATT	ERROR	ATTRIBUTE
ATTRIBUTE <ac-name> CAN NOT BE CREATED	BLDATT	ERROR	ATTRIBUTE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ATTRIBUTE <ac-name> IS ASSOCIATED WITH DOMAIN	VERACDT	WARNING	ATTRIBUTE
ATTRIBUTE <obj-id> DOES NOT EXIST	CRTALI	ERROR	ATTRIBUTE
ATTRIBUTE ALIAS NAME <AC-NAME> CAN NOT BE CREATED	WRTALI	ERROR	ATTRIBUTE
ATTRIBUTE COULD NOT BE BUILT	ICOPATT	ERROR	ATTRIBUTE
ATTRIBUTE COULD NOT BE COPIED	TERATTR	ERROR	ATTRIBUTE
ATTRIBUTE DESCRIPTION NOT CREATED	BLDATT1	ERROR	ATTRIBUTE
ATTRIBUTE DESCRIPTION NOT INSERTED	ICOPATT	ERROR	ATTRIBUTE
ATTRIBUTE KEYWORD CAN NOT BE CREATED	WRTACKW	WARNING	ATTRIBUTE
ATTRIBUTE NAME <ac-name> CAN NOT BE CREATED	WRTANAM	WARNING	ATTRIBUTE
ATTRIBUTE NAME TO BE COPIED AS MUST BE ENTERED	COPATT	ERROR	ATTRIBUTE
ATTRIBUTE NAMES SHOULD BE DIFFERENT	COPATT	ERROR	ATTRIBUTE
ATTRIBUTE USE CL NOT DELETED FOR ATTRIBUTE CLASS <attribute class number>	DELAUC	ERROR	ATTRIBUTE
ATTRIBUTE USE CLASS CAN NOT BE DELETED:	FNDAUC	ERROR	ATTRIBUTE
ATTRIBUTE USE CLASS WAS NOT DELETED	DELOAC	ERROR	ATTRIBUTE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
AUC ALG PARMS NOT FOUND FOR MODULE <MOD-ID> <MOD-INST-TEMP>	WRTALG	ERROR	CSIS/CSES
AUC MUST MAP TO SAME OWNER IN THE SAME DATABASE	VOMAPS	ERROR	CSIS
AUC TO FIELD MAPPING EXISTS	DELFLDS	ERROR	CSIS
AUC TO SET MAPPING EXISTS	DELFLDS	ERROR	CSIS
AUC VALUE <AUC-VALUE> IS NOT UNIQUE FOR THIS PREFERENCE	MAPASET	ERROR	ATTRIBUTE
AUC VALUE <AUC-VALUE> IS NOT UNIQUE FOR THIS PREFERENCE	ALTSMAP	ERROR	ATTRIBUTE
AUP TABLE OVERFLOW ERROR	SELAPRM	ERROR	
AUP TABLE OVERFLOW ERROR	SELAPRM	ERROR	
BAD CLOSE <SHOW-RC>	CHKKEYS	ERROR	MODEL
BAD DATA IN ATTRIBUTE_USE_CLASS	WRTWHCL	ERROR	VIEW
BAD FETCH <SHOW-RC>	CHKKEYS	ERROR	MODEL
BAD SELECT <SHOW-RC>	CHKKEYS	ERROR	MODEL
BEG. VALUE = <BEG-VALUE> BUT NO ENDING VALUE	ALTVLRG	WARNING	DOMAIN
BEG. VALUE = <DOM-BEG-VALUE> BUT NO ENDING VAL.	CRTVLRG	ERROR	DOMAIN
BEG. VALUE = <DOM-BEG-VALUE> BUT NO ENDING VALUE IN RANGE LIST	CRTVMOD	ERROR	DOMAIN
BIND 12 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND1 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
BIND10 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND11 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND13 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND14 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND15 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND2 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND3 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND4 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND5 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND6 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND7 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND8 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
BIND9 ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
CANNOT CREATE FILE FOR DESCRIPTION EDITING	WRDSFL	ERROR	
CANNOT DROP ATTRIBUTE <TAG-NAME>, IT IS NOT A MEMBER OF THE KEY CLASS	KCMDRP	ERROR	ATTRIBUTE
CANNOT OPEN PARCL1 <OPN-NAME>	CDP12	ERROR	
CANNOT OPEN PARCL2 <OPN-NAME>	CDP12	ERROR	
CANNOT OPEN PARCL3 <OPN-NAME>	CDP12	ERROR	
CANNOT OPEN PARCL4 <OPN-NAME>	CDP12	ERROR	

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
CANNOT SPECIFY VARIABLE AS LENGTH ZERO	ADDDT	ERROR	DATAITEM
COBINDN1 ERROR <MOD-NAME>	DELFLDA	ERROR	RECORD
COBINDN1 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN10 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN11 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN12 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN2 ERROR	DELFLDA	ERROR	RECORD
COBINDN2 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN3 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN4 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN5 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN6 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN7 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN8 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COBINDN9 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
CODASYL DATA BASE COULD NOT BE INSERTED	DEFCODL	ERROR	DATABASE
CODASYL REQUIRES A DATABASE LOCATION	DEFCODL	ERROR	DATABASE
CODASYL REQUIRES SCHEMA, SUBSCHEMA,AREA CLAUSE	DEFCODL	ERROR	DATABASE
COEXEC ERROR	DELFLDA	ERROR	RECORD
COEXEC ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
COMMAND NOT IMPLEMENTED	BRANCHR	ERROR	SYNTAX
COMMAND NOT PROCESSED	PROCMD	ERROR	
COMMAND UNRECOGNIZED	BRANCHR	ERROR	SYNTAX
COMPLETE RELATION NOT DELETED FOR KEY CLASS <KC-NO>	DELMTKC	ERROR	KEY
COMPLEX MAPPING EXISTS FOR TAG <TAG-NO>	VERTMAP	ERROR	CSES
COMPLEX MAPPING EXISTS FOR THIS TAG	DELMTAG	ERROR	CSIS
COMPONENT DATA FIELD IS INVALID FOR DATA FIELD <DF-ID>	RECSEMS	ERROR	RECORD
CONSTANT ALG PARMS NOT SELECTED FOR MODULE <MOD-ID> <MOD-INST-TEMP>	WRTALG	ERROR	CSIS/CSES
COPY VIEW 'ALL' NOT COMPLETED	CPYVIEW	ERROR	VIEW
COPY VIEW <VIEW-ID> NOT COMPLETED	CPYVIEW	ERROR	VIEW
COSQL3 ERROR <MOD-NAME>	DELFLDA	ERROR	RECORD
COSQL3 ERROR <MOD-NAME>	MODFLD	ERROR	DATAFIELD
COULD NOT DELETE OLD DESCRIPTION TEXT	STRINS	ERROR	DESCRIPTION
COULD NOT DELETE OLD DESCRIPTION TEXT	FILEINS	ERROR	DESCRIPTION

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
COULD NOT LOCATE GROUP DATA FIELD FOR COMPONENT <DFT-DFID(DFT-INDEX)	ADDFLDS	ERROR	RECORD
COULD NOT VERIFY DATA BASE (DB-NAME)	CPYMSET	ERROR	
COULD NOT VERIFY SET ID (SET ID)	CPYMSET	ERROR	
CS-IS MAPPING NOT FOUND FOR THIS PREFERENCE.	DRPMAP	ERROR	CSIS
CS/ES MAPPING EXISTS FOR TAG <TAG-NO>	VERTMAP	ERROR	CSES
CS/IS MAPPING EXISTS FOR TAG <TAG-NO>	VERTMAP	ERROR	CSIS
DANGER... -PERMISSION DENIED TO DROP CDM	DRPDB	ERROR	DATABASE
DATA BASE <DB-ID> (CONTINUED ON NEXT MESSAGE)	ALTPART	ERROR	DATABASE
DATA BASE <DB-ID> (CONTINUED ON NEXT MESSAGE)	CRTPART	ERROR	DATABASE
DATA BASE <DB-ID> AND RECORD DO NOT EXIST ON CDM	VALDROP	ERROR	DATABASE
DATA BASE <DB-ID> IS INVALID	DRPDB	ERROR	DATABASE
DATA BASE <DB-ID> NOT DELETED FROM DATA-BASE	DRPDB	ERROR	DATABASE
DATA BASE <DB-ID> NOT DELETED FROM DATA-BASE-AREA	DRPDB	ERROR	DATABASE
DATA BASE <DB-ID> NOT DELETED FROM DATA-FIELD	DRPDB	ERROR	DATABASE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
DATA BASE <DB-ID> NOT DELETED FROM DB-AREA-ASSIGNMENT	DRPDB	ERROR	DATABASE
DATA BASE <DB-ID> NOT DELETED FROM DF-SET-LINKAGE	DRPDB	ERROR	DATABASE
DATA BASE <DB-ID> NOT DELETED FROM SCHEMA-NAMES	DRPDB	ERROR	DATABASE
DATA BASE <DB-ID> NOT DELETED FROM SET-TYPE-MEMBER	DRPDB	ERROR	DATABASE
DATA BASE <DB-NAME> ALREADY EXISTS	DEFDB	ERROR	DATABASE
DATA BASE <DB-NAME> COULD NOT BE INSERTED	DEFDB	ERROR	DATABASE
DATA BASE <DB-NAME> DOES NOT EXIST	ALTDB	ERROR	DATABASE
DATA BASE <DB-NAME> NOT DELETE FROM DB_PASSWORD	DRPDB	ERROR	DATA BASE
DATA BASE <DB-NAME> NOT DELETED FROM DESC-TEXT	DRPDB	ERROR	DATABASE
DATA BASE <DB_ID> NOT DELETED FROM RECORD-TYPE	DRPDB	ERROR	DATABASE
DATA BASE <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	DATABASE
DATA BASE IS INVALID	DRPSET	ERROR	DATABASE
DATA BASE NAME <DB-NAME> IS INVALID	DRPFLD	ERROR	DATAFIELD
DATA BASE NAME IS UNAVAILABLE	DRPFLD	ERROR	DATAFIELD
DATA BASE NAME NOT FOUND	GETDRT	ERROR	DATABASE
DATA BASE NAME NOT FOUND ON DATA_BASE	DBLOOK	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
DATA FIELD <DF-ID> COULD NOT BE INSERTED	DEFFLD	ERROR	DATAFIELD
DATA FIELD <DF-ID> COULD NOT BE INSERTED	DEFDF	ERROR	DATAFIELD
DATA FIELD <DF-ID> IS USED BY A SOFTWARE MODULE (2)	DELFLDS	ERROR	DATAFIELD
DATA FIELD <DF-ID> IS USED IN A SOFTWARE MODULE (2)	DELDBDF	ERROR	DATAFIELD
DATA FIELD <DF-ID> NOT DELETED FROM DATA_FIELD	DELFLDS	ERROR	DATAFIELD
DATA FIELD <DF-ID> NOT DELETED FROM DF-SET-LINKAGE	DELFLDS	ERROR	DATAFIELD
DATA FIELD <DF-ID> NOT DELETED FROM RECORD-SET	DELFLDS	ERROR	DATAFIELD
DATA FIELD <DF-ID> NOT DELETED FROM SET-TYPE-MEMBER	DELFLDS	ERROR	DATAFIELD
DATA FIELD <data field name> DOES NOT EXIST	MAPADF	ERROR	DATAFIELD
DATA FIELD <df-id> IS ASSOCIATED WITH DATA TYPE	VERDFDT	WARNING	DATAFIELD
DATA FIELD NOT ASSOCIATED WITH DB AND REC-TYPE	ADDUNIN	ERROR	CSIS
DATA FIELD NOT DELETED FROM DATA FIELD	DELDBDF	ERROR	DATAFIELD
DATA FIELDS CANNOT BE ADDED AFTER/BEFORE A FILLER	ADDFLDS	ERROR	RECORD
DATA FIELDS MUST BE KNOWN TO CDM FOR RELATIONAL DATA BASES	FLDSEMS	ERROR	RECORD
DATA ITEM <di-id> IS ASSOCIATED WITH DATATYPE	VERDIDT	WARNING	USERTYPE
DATA ITEM ALGPARMS NOT FOUND FOR MODULE <MOD-ID> <MOD-INST-TEMP>	WRTALG	ERROR	CSIS/CSES

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
DATA ITEMS CANNOT BE SPECIFIED WITH 'SELECT *' CLAUSE	BLVWLST	ERRROR	VIEW
DATA TYPE COULD NOT BE ADDED FOR <DATA-TYPE-NAME> TYPE : <STD-USER>	ADDDT	ERROR	USERTYPE
DATA TYPE COULD NOT BE INSERTED <data-type-name>	ADDNSTD	ERROR	USERTYPE
DATA TYPE COULD NOT BE UPDATED TO STD FOR: <DATA-TYPE-NAME>	ALTDT	ERROR	USERTYPE
DATA TYPE COULDN'T BE UPDATED FOR : <DATA-TYPE-NAME>	ALTDT	ERROR	USERTYPE
DATA TYPE IS 'STD' - SHOULD NOT BE DROPPED: <DATA-TYPE-NAME>	DRPDT	ERROR	USERTYPE
DATA TYPE NAME ALREADY EXISTS <data-type-name>	ADDNSTD	ERROR	USERTYPE
DATA TYPE NAME COULD NOT BE FOUND	VMODPRM	ERROR	DOMAIN
DATA TYPE NAME DOESN'T EXIST	DEFSMOD	ERROR	MODULE
DATA TYPE NAME DOESN'T EXIST <DATA-TYPE-NAME>	ADDPRMS	ERROR	MODULE
DATA TYPE NAME IS NOT IN DOMAIN	DRPDT	ERROR	USERTYPE
DATA TYPES COULD NOT BE DELETED FOR DOMAIN : <DOM-NAME>	DRPDOM	ERROR	DOMAIN
DATA TYPES COULD NOT BE RETRIEVED FOR <DOM-NAME>	DRPDOM	ERROR	DOMAIN
DATA TYPES NOT COMPATIABLE FOR TAG	BLVWLST	ERROR	VIEW
DATA-FIELD <DF-ID> DOES NOT EXIST	SELFLD	ERROR	DATAFIELD

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
DATA-TYPE COULD NOT BE DELETED <data-type-name>	DRPDT	ERROR	USERTYPE
DATA-TYPE COULD NOT BE INSERTED : <DATA-TYPE-NAME>	ADDSTD	ERROR	USERTYPE
DATA-TYPE NAME NOT IN DOMAIN - <DATA-TYPE-NAME>	ALTDT	ERROR	USERTYPE
DATA-TYPE-NAME ALREADY EXISTS : <DATA-TYPE-NAME>	ADDSTD	ERROR	USERTYPE
DATABASE (DB-NAME) DOESN'T EXIST	CPYSET	ERROR	CSIS
DATABASE <DB-NAME> DOES NOT EXIST	MAPAREC	ERROR	CSIS
DATABASE <OBJ-ID-1> DOES NOT EXIST	VEROBJ	ERROR	DATABASE
DATABASE <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	DATABASE
DATABASE AND RECORD ID NOT FOUND	ALGCHK	ERROR	CSIS
DATABASE CDM CANNOT BE RENAMED	RENMOBJ	ERROR	DATABASE
DATABASE ERROR - UNABLE TO ADD DESCRIPTION TEXT	FILEINS	ERROR	DESCRIPTION
DATABASE ERROR - UNABLE TO ADD DESCRIPTION TEXT	STRINS	ERROR	DESCRIPTION
DATABASE NAME - <DB-NAME> DOES NOT EXIST	DRPSMAP	ERROR	DATABASE
DATABASE NAME <DB-NAME> DOES NOT EXIST	ALTSMAP	ERROR	DATABASE
DATABASE NAME <database name> DOES NOT EXIST	ADDMAP	ERROR	DATABASE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
DATAFIELD <DF-ID> DOES NOT EXIST	VEROBJ	ERROR	DATAFIELD
DATAFIELD <DF-NAME> DOES NOT EXIST	ADDMAP	ERROR	DATAFIELD
DATAFIELD <DF-NAME> DOES NOT EXIST	DRPSMAP	ERROR	DATAFIELD
DATAFIELD <DF_ID> DOES NOT EXIST	VEROBJ1	ERROR	DATAFIELD
DATAITEM <DI-ID> DOES NOT EXIST	VEROBJ	ERROR	VIEW
DATAITEM PARTICIPATES IN AN ALGORITHM	DRPDIV	ERROR	VIEW
DB <OBJ-ID-1> DOES NOT EXIST	VEROBJ	ERROR	DATABASE
DB <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	DATABASE
DBMS (DBMS-NAME) DOES NOT EXIST	CPYDBMS	ERROR	
DBMS / HOST ASSOCIATION NOT *DEFINED*	ALTDBMS	ERROR	DBMS
DBMS / HOST ASSOCIATION NOT DEFINED	ALTHST	ERROR	DBMS
DBMS <DBMS-NAME> DOES NOT EXIST	DEFDB	ERROR	DATABASE
DBMS <DBMS-NAME> IS NOT SUPPORTED IN CURRENT CDM	ALTDB	ERROR	DATABASE
DBMS <DBMS-NAME> NOT DEFINED ON DBMS TABLE	ALTDBMS	ERROR	DBMS
DBMS <DBMS-NAME> NOT DEFINED	ALTHST	ERROR	DBMS
DBMS <DBMS-NAME> NOT DEFINED	DEFHST	ERROR	DBMS
DBMS <DBMS-NAME> NOT ON DBMS TABLE	DRPDBMS	ERROR	DBMS

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
DBMS <DBMS-NAME> NOT SUPPORTED IN CURRENT CDM	DEFDB	ERROR	DATABASE
DBMS OF DATABASE (DB-NAME) IS ORACLE - SETS ARE ILLEGAL	CPYSET	ERROR	
DECIMAL SPECIFICATION INCORRECT FOR: <DATA-TYPE-NAME>	PROCDT	ERROR	
DEFINE ALGORITHM COMMAND NOT GENERATED	GENVIEW	ERROR	VIEW
DEFINE SET IS ILLEGAL FOR <DBMS-NAME>	DEFSET	ERROR	DATABASE
DELETE LIST FOR : <MOD-ID>	DRPSMOD	WARNING	MODULE
DEPENDENT ENTITY <DF-NAME> DOES NOT EXIST IN MODEL	TERSTR	ERROR	ENTITY
DEPENDENT ENTITY <DF-NAME> DOES NOT EXIST IN THE MODEL	TERREL	ERROR	ENTITY
DEPENDENT ENTITY NAME (DEP-EC-NAME) IS INVALID	VERREL	ERROR	
DEPENDENT ENTITY NAME <DEP-EC-NAME> IS INVALID	VEREL	ERROR	ENTITY
DEPENDENT ENTITY NAME <DEP-EC-NAME> IS INVALID	VERREL	ERROR	ENTITY
DEPENDENT ENTITY TABLE COULD NOT BE CONSTRUCTED	CPYMOD	ERROR	
DEPENDING DATA FIELD MUST BE NUMERIC DATA-TYPE. SEE FIELD - <DF-ID>	RECSEMS	ERROR	RECORD
DEPENDING ON DATA FIELD <DF-ID> NOT DEFINED	DEFFLDS	ERROR	RECORD
DEPENDING ON DATA FIELD CANNOT BE GROUP OR REPEATING OR INDEX. SEE FIELD - <DF-ID>	RECSEMS	ERROR	RECORD

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
DEPENDING ON FIELD <FIELD-NAME-2> DOES NOT EXIST	ALTFLDS	ERROR	DATAFIELD
DESC TEXT NOT DELETED FOR <VIEW-NAME>	DRPVIEW	ERROR	VIEW
DESCRIPTION <DESC-TYPE> NOT ON CDM.	DRPDSCT	ERROR	DESCRIPTION
DESCRIPTION TEXT DOES NOT EXIST FOR DESCRIPTION TYPE <DESC-TYPE>	CPDDRSP	ERROR	DESCRIPTION
DESCRIPTION TEXT DOES NOT EXIST FOR DESCRIPTION TYPE <DESC-TYPE>	CPDOFSP	ERROR	DESCRIPTION
DESCRIPTION TEXT FOR SET NOT DELETED	SELRSET	ERROR	RECORD
DESCRIPTION TEXT NOT DELETED FOR SET	SELSTM	ERROR	RECORD
DESCRIPTION TYPE <DESC-TYPE> ALREADY ON CDM	CRTDSCT	ERROR	DESCRIPTION
DESCRIPTION TYPE <DESC-TYPE> COULD NOT BE DELETED.	DRPDSCT	ERROR	DESCRIPTION
DESCRIPTION TYPE <DESCRIPTION TYPE> DOES NOT EXIST.	DESCRB	ERROR	DESCRIPTION
DESCRIPTIONS CAN NOT BE DROPPED FOR ENTITY <ec-name>	DRPENT	ERROR	ENTITY
DF-TABLE OVERFLOWED	DRPFLD	ERROR	DATAFIELD
DFP TABLE OVERFLOW ERROR	SELDFPM	ERROR	
DFT-TABLE OVERFLOW	SELFLD	ERROR	DATAFIELD
DISTRIBUTED RULES ALREADY EXIST FOR ENTITY <EC_NO>	CRTMAP	ERROR	ENTITY
DOMAIN (DOMAIN_NAME) NOT FOUND	CPYDOM	ERROR	
DOMAIN <DOM-NAME> IS ASSOCIATED WITH AN ATTRIBUTE	DRPDOM	ERROR	DOMAIN
DOMAIN <OBJ-ID-1> DOES NOT EXIST	VEROBJ	ERROR	DOMAIN

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
DOMAIN <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	DOMAIN
DOMAIN <domain-name> ALREADY EXISTS	CRTATT	ERROR	DOMAIN
DOMAIN ALREADY EXISTS - <dom-name>	CRTDOM	ERROR	DOMAIN
DOMAIN COULD NOT BE DELETED FOR : <DOM-NAME>	DRPDOM	ERROR	DOMAIN
DOMAIN COULD NOT BE INSERTED - <dom-name>	CRTDOM	ERROR	DOMAIN
DOMAIN DOES NOT EXIST FOR : <DOM-NAME>	DRPDOM	ERROR	DOMAIN
DOMAIN IN USE CAN'T BE DROPPED FOR : <DOM-NAME>	DRPDOM	ERROR	DOMAIN
DOMAIN NOT CHANGED FOR ATT-CL <attribute classs name>	ALTATT	ERROR	DOMAIN
DOMAIN NUMBER OVERFLOW	CRTDOM	ERROR	DOMAIN
DOMAIN: <DOMAIN-NAME> NOT FOUND	CRTATT	ERROR	DOMAIN
DUPLICATED ENTITY NAMES IN FROM LIST	BLVWLST	ERROR	VIEW
DURING BUILDING OF DEP-EC-LST	TERREL	ERROR	ENTITY
DURING OPENING OF FILE	TERREL	ERROR	DATABASE
DURING OPENING OF FILE	TERSIM	ERROR	DATABASE
EC KEYWORD ALREADY EXISTS	ADDKWE	ERROR	KEYWORD
EDIT ERRORS FOUND FOR <DOM-NAME>	ADDSTD	ERROR	USERTYPE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ELEMENTARY ITEMS MUST HAVE A DATA TYPE NAME	RECSEMS	ERROR	DATAITEM
END OF INPUT ENCOUNTERED	PROCMD	ERROR	SYNTAX
END OF INPUT ENCOUNTERED	PROCMD	ERROR	
END OF INPUT ENCOUNTERED DESCRIBE NOT PERFORMED	PROCMD	ERROR	
ENTITIES IN WHERE CLAUSE MUST BE SPECIFIED IN	BLVWLST	ERROR	VIEW
ENTITY <EC-NAME> CAN NOT BE CREATED	CRTENT	ERROR	ENTITY
ENTITY <EC-NAME> COULD NOT BE DELETED	GETECAL	ERROR	ENTITY
ENTITY <EC-NAME> DOES NOT EXIST	ALTALI	ERROR	ENTITY
ENTITY <EC-NAME> DOES NOT EXIST	BLVWLST	ERROR	VIEW
ENTITY <EC-NAME> DOES NOT EXIST	CRTPART	ERROR	ENTITY
ENTITY <EC-NAME> DOES NOT EXIST	HPNLOOK	ERROR	ENTITY
ENTITY <EC-NAME> DOESN'T EXIST IN RECORD UNION	ALTUNIN	ERROR	CSIS
ENTITY <EC-NAME> IS HORIZONTALLY PARTITIONED	DRPENT	ERROR	ENTITY
ENTITY <EC-NAME> NOT INSERTED INTO ENTITY-CLASS	ICOPENT	ERROR	ENTITY
ENTITY <EC-NAME> NOT INSERTED INTO ENTITY-NAME	ICOPENT	ERROR	ENTITY

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ENTITY <EC-NAME> SYSTEM ERROR : UNABLE TO ALTER ALIAS	ALTALI	ERROR	ENTITY
ENTITY <IND-EC-NAME> DOES NOT EXIST IN THE CDM	ALTMAP	ERROR	ENTITY
ENTITY <IND-EC-NAME> DOES NOT EXIST IN THE CDM	CRTMAP	ERROR	CSIS
ENTITY <OBJ-ID-1> DOES NOT EXIST	VEROBJ	ERROR	ENTITY
ENTITY <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	ENTITY
ENTITY <TO-EC-NAME> NOT INSERTED	TRADIR	ERROR	ENTITY
ENTITY <ec-name> COULD NOT BE DROPPED	DRPENT	ERROR	ENTITY
ENTITY <entity name> DOES NOT EXIST AND CAN NOT BE DROPPED	DRPENT	ERROR	ENTITY
ENTITY <obj-id> DOES NOT EXIST	CRTALI	ERROR	ENTITY
ENTITY ABBREVIATIONS REQUIRED IN WHERE CLAUSE	BLVWLST	ERROR	VIEW
ENTITY AND RECORD TYPE ALREADY EXIST IN CDM	CRTPART	ERROR	ENTITY
ENTITY CLASS <EC-NAME> DOES NOT EXIST	ALTMAP	ERROR	ENTITY
ENTITY CLASS <entity class name> DOES NOT EXIST	CRTMAP	ERROR	ENTITY
ENTITY CLASS <entity class name> DOES NOT EXIST	DRPMAP	ERROR	ENTITY
ENTITY CONSTRAINTS FOR <EC-NAME> COULD NOT BE DROPPED	DRPENT	ERROR	ENTITY

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
ENTITY HAS BEEN PREVIOUSLY MAPPED FOR THIS PREFERENCE TO A NON-HP RECORD	ADDMAP	ERROR	CSIS
ENTITY KEYWORD CAN NOT BE CREATED	WRTECKW	WARNING	ENTITY
ENTITY MISSING FROM EC-NAME-TABLE	WRTWHCL	ERROR	VIEW
ENTITY MUST HAVE AT LEAST TWO HORIZONTAL PART	ALTPART	ERROR	ENTITY
ENTITY NAME <EC-NAME> DOES NOT EXIST	ALTENT	ERROR	ENTITY
ENTITY NAME <EC-NAME> EXISTS	CRTENT	ERROR	ENTITY
ENTITY NAME <ec-name> CAN NOT BE CREATED	WRTENAM	ERROR	ENTITY
ENTITY NAME DOES NOT EXIST IN THE CDM	ALGCHK	ERROR	CSIS
ENTITY NAMES MUST DIFFER ON INTRA-MODEL COMBINE	CMBENT	ERROR	ENTITY
ENTITY NAMES MUST DIFFER ON INTRA-MODEL COMBINE	CMBENT	ERROR	ENTITY
ENTITY TABLE OVERFLOW - OVER 25 DROPPED ON ALTER ENTITY COMMAND	ALTUNIN	ERROR	CSIS
ENTITY TO RECORD MAPPING ALREADY EXISTS	MAPAREC	ERROR	CSIS
ENTITY TO RECORD MAPPING DOES NOT EXIST & CAN'T BE DELETED	DRPSMAP	ERROR	CSIS
ENTITY TO RECORD MAPPING EXISTS	DRPDB	ERROR	DATABASE
ERROR ALTERING DATA BASE <DB-NAME>	ALTDB	ERROR	DATABASE
ERROR CHANGING ATTRIBUTE KEYWORD NUMBER	RENMOBJ	ERROR	KEYWORD

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
ERROR CHANGING DB-MODEL	ALTDBMS	ERROR	DATABASE
ERROR CHANGING ENTITY KEYWORD NUMBER	RENMOBJ	ERROR	KEYWORD
ERROR CHANGING RELATION KEYWORD NUMBER	RENMOBJ	ERROR	KEYWORD
ERROR DELETING AREA <AREA-NAME>	ALTCODL	ERROR	DATABASE
ERROR DELETING DBMS / HOST ASSOCIATION	ALTHST	ERROR	DBMS
ERROR DELETING DBMS <DBMS-NAME>	DRPDBMS	ERROR	DBMS
ERROR DELETING FILE <FILE-NAME>	ALTTOT	ERROR	DATABASE
ERROR DELETING FROM AUC_IS_MAPPING	DRPALG	ERROR	CSIS/CSES
ERROR DELETING FROM CDM TABLE KEY CLASS MEMBER	PRESMIG	ERROR	KEY
ERROR DELETING FROM CDM TABLE KEY_CLASS_MEMBER	AOADPM	ERROR	KEY
ERROR DELETING HOST / DBMS ASSOCIATION	ALTDBMS	ERROR	DBMS
ERROR DELETING HOST <HOST-ID>	DRPHST	ERROR	HOST
ERROR DELETING OLD ATTRIBUTE KEY MIGRATIONS	AOADPM	ERROR	ATTRIBUTE
ERROR DELETING OLD KEYWORD <OLD-ID>	RENMOBJ	ERROR	KEYWORD
ERROR DROPPING ALL ATTRIBUTE TO FIELD MAPPINGS FOR ENTITY	DRPMAP	ERROR	CSIS
ERROR DROPPING THE ATTRIBUTE TO DATAFIELD/SET MAPPING	SELAIMP	ERROR	CSIS
ERROR DROPPING THE ENTITY TO RECORD MAP	DRPALTG	ERROR	CSIS
ERROR DROPPING THE ENTITY TO RECORD MAP	DRPPRF1	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
ERROR DROPPING THE ENTITY TO RECORD MAPPING	SELAIMP	ERROR	CSIS
ERROR DURING INSERT INTO ATTRIBUTE USE CLASS	TERATTR	ERROR	ATTRIBUTE
ERROR DURING VERIFYING RELATION, RC-NAME : <RC-NAME>	VERRELS	ERROR	RELATION
ERROR ENCOUNTERED DURING INTEGRITY CHECKS	ADDFLDS	ERROR	RECORD
ERROR ENCOUNTERED DURING INTEGRITY CHECKS	ALTFLDS	ERROR	DATAFIELD
ERROR ENCOUNTERED DURING VERIFICATION OF ENTITY/PARTITION	ALTPART	ERROR	ENTITY
ERROR ENCOUNTERED VERIFYING PARTITION OR ENTITY	DRPPART	ERROR	ENTITY
ERROR ENCOUNTERED WHILE DROPPING DATA BASE/RECORD TYPES	ALTPART	ERROR	ENTITY
ERROR GENERATING CREATE ENTITY COMMAND	CERELS	ERROR	ENTITY
ERROR GETTING GLOBAL PARAMETERS FOR CURRENT DB	CPYSET	ERROR	
ERROR IN BUILDING DEP-EC-LST	TERSTR	ERROR	ENTITY
ERROR IN DEFINING DATA BASE	DEFDB	ERROR	DATABASE
ERROR IN DELETING INHERITED ATTRIBUTE USE CLASSES	PRESMIG	ERROR	TAG
ERROR IN DELETING KEY CLASS MIGRATIONS	PRESMIG	ERROR	KEY
ERROR IN DELETING TEXTUAL DESCRIPTION FOR DATAITEM	DRPDIV	ERROR	VIEW
ERROR IN DETERMINING COMPLEX MAPPING	DRPVIEW	ERROR	VIEW
ERROR IN INSERTING ATTRIBUTE USE CLASS OCCURENCE	AOAPICR	ERROR	TAG

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
ERROR IN INSERTING KEY CLASS MEMBER AS NEW OWNER	AOAPICR	ERROR	KEY
ERROR IN MAKING RELATION CLASS COMPLETE	AOAPICR	ERROR	RELATION
ERROR IN NDML INSERT	SEL1DSC	ERROR	DESCRIPTION
ERROR IN NDML SELECT	BLVWLST	ERROR	VIEW
ERROR IN OBINDN	RETRFLD	ERROR	RECORD
ERROR IN OBINDN	SELFLD	ERROR	DATAFIELD
ERROR IN OBTAINING NEW TAG NUMBER	TERATTR	ERROR	TAG
ERROR IN ODFINN1	RETRFLD	ERROR	RECORD
ERROR IN ODFINN1	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN10	RETRFLD	ERROR	RECORD
ERROR IN ODFINN10	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN11	RETRFLD	ERROR	RECORD
ERROR IN ODFINN11	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN12	RETRFLD	ERROR	RECORD
ERROR IN ODFINN12	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN13	RETRFLD	ERROR	RECORD
ERROR IN ODFINN13	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN14	RETRFLD	ERROR	RECORD
ERROR IN ODFINN14	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN2	RETRFLD	ERROR	RECORD
ERROR IN ODFINN2	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN3	RETRFLD	ERROR	RECORD
ERROR IN ODFINN3	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN4	RETRFLD	ERROR	RECORD
ERROR IN ODFINN4	SELFLD	ERROR	DATAFIELD

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
ERROR IN ODFINN5	RETRFLD	ERROR	RECORD
ERROR IN ODFINN5	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN6	RETRFLD	ERROR	RECORD
ERROR IN ODFINN6	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN7	RETRFLD	ERROR	RECORD
ERROR IN ODFINN7	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN8	RETRFLD	ERROR	RECORD
ERROR IN ODFINN8	SELFLD	ERROR	DATAFIELD
ERROR IN ODFINN9	RETRFLD	ERROR	RECORD
ERROR IN ODFINN9	SELFLD	ERROR	DATAFIELD
ERROR IN OEXEC	RETRFLD	ERROR	RECORD
ERROR IN OEXEC	SELFLD	ERROR	DATAFIELD
ERROR IN OFETCH	SELFLD	ERROR	DATAFIELD
ERROR IN OFETCH	RETRFLD	ERROR	RECORD
ERROR IN OOPEN	RETRFLD	ERROR	RECORD
ERROR IN OOPEN	SELFLD	ERROR	DATAFIELD
ERROR IN OPENING FILE	CPDOFSP	ERROR	DESCRIPTION
ERROR IN OPENING FILE	CPDOFAL	ERROR	DESCRIPTION
ERROR IN ORACLE LOGON	INITSES	ERROR	
ERROR IN OSQL3	RETRFLD	ERROR	RECORD
ERROR IN OSQL3	SELFLD	ERROR	DATAFIELD
ERROR IN SELECTING FROM CDM	CPDDRAL	ERROR	DESCRIPTION
ERROR IN SELECTING VIEW	CPYVIEW	ERROR	VIEW
ERROR IN VALIDATING JOIN CONDITIONS	REMVIEW	ERROR	VIEW
ERROR IN VERIFYING KEY CLASS MIGRATIONS	MKRNLST	ERROR	RELATION

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ERROR INSERTING <TAG-NAME-WS> INTO ATTRIBUTE USE CLASS	AOAGAI	ERROR	TAG
ERROR INSERTING <TAG-NAME-WS> INTO INHERITED ATTRIBUTE USE CLASS	AOAGAI	ERROR	TAG
ERROR INSERTING A KEY CLASS MEMBER OCCURENCE	AOAMGDP	ERROR	KEY
ERROR INSERTING AREA <AREA-NAME>	ALTCODL	ERROR	DATABASE
ERROR INSERTING ATTRIBUTE USE CLASS OCCURENCE	AOAMGDP	ERROR	TAG
ERROR INSERTING DBMS / HOST ASSOCIATION	DEFDBMS	ERROR	DBMS
ERROR INSERTING DBMS / HOST ASSOCIATION	ALTHST	ERROR	DBMS
ERROR INSERTING DBMS / HOST ASSOCIATION	DEFHST	ERROR	DBMS
ERROR INSERTING DBMS <DBMS-NAME>	DEFDBMS	ERROR	DBMS
ERROR INSERTING DESCRIPTION TYPE <DESC-TYPE>	CRTDSCT	ERROR	DESCRIPTION
ERROR INSERTING FILE <FILE-NAME>	ALTTOT	ERROR	DATABASE
ERROR INSERTING HOST / DBMS ASSOCIATION	ALTDBMS	ERROR	DBMS
ERROR INSERTING HOST <HOST-ID>	DEFHST	ERROR	HOST
ERROR INSERTING INHERITED ATTRIBUTE USE CLASS	AOAPICR	ERROR	TAG

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ERROR INSERTING INHERITED ATTRIBUTE USE OCCURENCE	AOAMGDP	ERROR	TAG
ERROR INSERTING INTO DESC_TEXT	SELDESC	ERROR	DESCRIPTION
ERROR INSERTING KEY CLASS MEMBER IN NEW OWNER	AOAEVAL	ERROR	KEY
ERROR INSETING INTO AUC-IS-MAPPING	ALGCHK	ERROR	CSIS
ERROR LOCATING INDEXING FIELD : <SEARCH-INDEX-DFNO>	DRPFLD	ERROR	DATAFIELD
ERROR OCCURED DURING INTEGRITY CHECKS - RECORD NOT INSERTED	DEFFLDS	ERROR	RECORD
ERROR OCCURED WHILE SELECTING HOST ASSOCIATION FOR DBMS	CPYDBMS	ERROR	
ERROR OCCURED WHILE SELECTING HOST ASSOCIATIONS FOR DBMS	ALLDBMS	ERROR	
ERROR RETRIEVING DATABASE AND RECORD ID	ALGCHK	ERROR	CSIS
ERROR SELECTING FROM DESC_TEXT	CPDDRSP	ERROR	DESCRIPTION
ERROR SELECTING FROM DESC_TEXT	CPDOFSP	ERROR	DESCRIPTION
ERROR SELECTING QUALIFICATION CRITERIA ITEMS	GENVIEW	ERROR	VIEW
ERROR UPDATING DATA BASE <DB-NAME>	ALTDB	ERROR	DATABASE
ERROR UPDATING DATA FIELD <OLD-ID>	UPDFLD	ERROR	DATAFIELD
ERROR UPDATING DATA ITEM <OLD-ID>	UPDDTI	ERROR	DATAITEM

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ERROR UPDATING DATA TYPE <OLD-ID>	UPDDT	ERROR	USERTYPE
ERROR UPDATING DB LOCATION <DB-LOCATION>	ALTCODL	ERROR	DATABASE
ERROR UPDATING HOST <OLD-ID>	UPDHOST	ERROR	HOST
ERROR UPDATING PASSWORD <DB-PASSWORD>	ALTORCL	ERROR	DATABASE
ERROR UPDATING PSB <PSB-NAME>	ALTIMS	ERROR	DATABASE
ERROR UPDATING RECORD TYPE <OLD-ID>	UPDREC	ERROR	RECORD
ERROR UPDATING SCHEMA <SCHEMA-NAME> SUBSCHEMA <SUBSCHEMA>	ALTCODL	ERROR	DATABASE
ERROR UPDATING SET <OLD-ID>	UPDSET	ERROR	SET
ERROR WHILE COUNTING THE ENTITY TO RECORD MAPPINGS	DRPALTG	ERROR	CSIS
ERROR WHILE COUNTING THE ENTITY TO RECORD MAPPINGS	DRPPRF1	ERROR	CSIS
ERROR WHILE COUNTING THE ENTITY TO RECORD MAP	DRPSPRF1	ERROR	CSIS
ERROR WHILE DELETING ATTRIBUTE <AC-NAME> FROM MODEL	FNDACM	ERROR	ATTRIBUTE
ERROR WHILE DELETING ENTITY <EC-NAME> FROM MODEL	FNDECM	ERROR	ENTITY
ERROR WHILE DELETING KEY CLASS FOR ENTITY <EC-NAME>	DELMDKC	ERROR	KEY
ERROR WHILE DELETING RELATION FOR ENTITY <EC-NAME>	DELMDRC	ERROR	RELATION
ERROR WHILE DELETING TAG FOR ENTITY <EC-NAME>	DLMDAUC	ERROR	ATTRIBUTE
ERROR WHILE DROPPING DISTRIBUTED RULES FOR ENTITY, <EC_NAME>	DRPMAP	ERROR	ENTITY
ERROR WHILE DROPPING MAP FOR TAG	DRPMAP	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
ERROR WHILE DROPPING THE ATTRIBUTE TO DATAFIELD/SET MAPPING	SELAIMT	ERROR	CSIS
ERROR WHILE DROPPING THE ENTITY DISTRIBUTED RULES	DRPSPRF	ERROR	ENTITY
ERROR WHILE DROPPING THE ENTITY'S DISTRIBUTED RULES	DRPALTG	ERROR	ENTITY
ERROR WHILE DROPPING THE ENTITY'S DISTRIBUTED RULES	DRPPRF1	ERROR	ENTITY
ERROR WHILE MERGING ENTITIES OF MODEL 2	MRGMOD	ERROR	MODEL
ERROR WHILE SELECTING ENTITY TO RECORD MAPPING FOR ENTITY <IND-EC-NAME>	DRPMAP	ERROR	CSIS
ERROR WHILE SELECTING THE ENTITY TO RECORD MAPPINGS	DRPSPRF	ERROR	CSIS
ERROR--ALIAS ALREADY EXISTS IN TO-MODEL	WRTENAM	ERROR	ENTITY
ERROR: DEP-EC-LIST HAS OVERFLOWED	LDECLST	ERROR	ENTITY
ERROR: KC-NAME <KC-NAME-LST> IS SPECIFIED BY USER MULTIPLE TIMES	AOAKC	ERROR	KEY
ERROR: NEW OWNER ENTITY <EC-NAME-NEW> DOES NOT EXIST	AOAEVAL	ERROR	ENTITY
ERROR: OLD AND NEW OWNER ENTITIES ARE BOTH DEPENDENT ON EACH OTHER-DEPENDENCY LOOP EXISTS	AOACHK	ERROR	ENTITY
ERROR: OLD OWNER MIGRATES MULTIPLE TIMES TO NEW OWNER, AND MORE THAN ONE INHERITED ATTRIBUTE IS KEY IN NEW OWNER	AOACHK	ERROR	ATTRIBUTE
ERROR:INCOMPLETE RELATION HAS NOT BEEN SPECIFIED	AOAPICR	ERROR	RELATION
ERROR:SPECIFIED RELATION CLASS <RC-NAME-LST> DOES NOT EXIST	AOAPICR	ERROR	RELATION

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ERRORS ENCOUNTERED ALL OTHER COMMANDS WILL BE SYNTAX CHECKED --- ANY CHANGES SINCE LAST COMMIT HAVE BEEN ROLLED BACK ---	PROCMD	WARNING	SYNTAX
ERRORS IN RELATION CLASS COMPARE	CMPMOD	ERROR	MODEL
EXCEPTED DEPENDENT ENTITY <DF-NAME> IS THE SAME AS THE ENTITY BEING COPIED	TERREL	ERROR	ENTITY
EXCEPTED DEPENDENT ENTITY <DF-NAME> IS THE SAME AS THE ENTITY BEING COPIED	TERSTR	ERROR	ENTITY
EXCEPTED ENTITY <DF-NAME> WAS AN INDEPENDENT ENTITY	TERREL	WARNING	ENTITY
EXCEPTION OF DEPENDENT ENTITY <DF-NAME> ILLEGAL	COPENT	ERROR	ENTITY
EXECUTE ERROR	INSFLD	ERROR	RECORD
EXPECTING STRUCTURE OR RELATIONS	FCOPENT	ERROR	
FIELD BELONGS IN A RECORD-UNION	DELFLDS	ERROR	CSIS
FIELD DEFINITIONS NOT ACCOMPLISHED	DEFREC	ERROR	RECORD
FIELD PARTICIPATES IN AN ALGORITHM	DELFLDS	ERROR	CSIS
FIELD THAT IS REDEFINING CANNOT BE REPEATING OR AN INDEX. SEE FIELD - <DF-ID>	RECSEMS	ERROR	RECORD
FILE <FILE-NAME> ALREADY EXISTS	DEFTOT	ERROR	DATABASE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
FILE <FILE-NAME> ALREADY EXISTS	ALTTOT	ERROR	DATABASE
FILE <FILE-NAME> DOES NOT EXIST	ALTTOT	ERROR	DATABASE
FILE <filename> HAS NO DATA	FILEINS	ERROR	DESCRIPTION
FILE CLAUSE REQUIRED FOR TOTAL DBMS	DEFTOT	ERROR	DATABASE
FILLER MUST ALWAYS BE UNKNOWN TO THE DBMS	ADDFLDS	ERROR	RECORD
FILLER MUST BE SPECIFIED AS UNKNOWN TO DBM	DEFFLDS	ERROR	RECORD
FILLER SIZE MUST CONTAIN A NUMERIC VALUE	ADDFLDS	ERROR	RECORD
FILLER SIZE SHOULD BE A NON-ZERO NUMERIC VALUE	ADDFLDS	ERROR	RECORD
FILLER-SIZE SHOULD BE A NON-ZERO NUMERIC VALUE	DEFFLDS	ERROR	RECORD
FROM CLAUSE ENTITY (EC-NAME) DOES NOT EXIST	CPYMENT	ERROR	ENTITY
FROM CLAUSE MISSING/INVALID	BLVWLST	ERROR	VIEW
FROM CLAUSE MUST BE SPECIFIED	BLVWLST	ERROR	VIEW
FROM-ENTITY <ENT-NAME> DOES NOT EXIST	CMBENT	ERROR	ENTITY
FROM-ENTITY NAME <FROM-EC-NAME> DOES NOT EXIST	COPENT	ERROR	ENTITY
FROM-MODEL <FROM-MODEL-NAME> DOES NOT EXIST	COPENT	ERROR	ENTITY

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
FROM-MODEL <MOD-NAME> DOES NOT EXIST	CMBENT	ERROR	ENTITY
FROM-MODEL-NAME AND CUR-MODEL-NAME CANNOT BE THE SAME	COPENT	ERROR	MODEL
GDATA ERROR IN PROCMD - RCODE IS <RCODE>	PROCMD	ERROR	
GLOBAL DATA BASE INFORMATION INCORRECT	GETRDH	ERROR	DATABASE
GLOBAL DATA BASE INFORMATION INCORRECT	GETDBST	ERROR	DATABASE
HORIZONTAL PARTITION ALREADY EXISTS ON CDM	ALTPART	ERROR	ENTITY
HORIZONTAL PARTITION COULD NOT BE DELETED	DRPPART	ERROR	ENTITY
HOST (HOST-ID) NOT DEFINED	CPYHST	ERROR	
HOST <HOST-ID> ALREADY ON HOST TABLE	DEFHST	ERROR	HOST
HOST <HOST-ID> DOES NOT EXIST	DEFDB	ERROR	HOST
HOST <HOST-ID> NOT DEFINED	ALTDB	ERROR	HOST
HOST <HOST-ID> NOT DEFINED ON HOST TABLE	ALTHST	ERROR	HOST

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
HOST <HOST-ID> NOT DEFINED	ALTDBMS	ERROR	HOST
HOST <HOST-ID> NOT DEFINED	DEFDBMS	ERROR	HOST
HOST <HOST-ID> NOT ON HOST TABLE	DRPHST	ERROR	HOST
HOST NAME <OBJ-ID-1> NOT DEFINED	VEROBJ	ERROR	HOST
HOST NAME <OBJ_ID_1> NOT DEFINED	VEROBJ1	ERROR	HOST
HP TABLE OVERFLOW ERROR	GENCRPT	ERROR	
I/O ERROR ON DESCRIPTION FILE <IFILE> STATUS CODE *<FILE-STAT>*	FILEINS	ERROR	DESCRIPTION
IAUC NOT DELETED FOR ATTRIBUTE CLASS	DELAUC	ERROR	TAG
IF GROUP LEVEL IS UNIQUE KEY AND SUBCOMPONENT IS KEY, IT MUST BE DUPLICATE KEY. SEE <TDFT-DFID(TDFT-INDEX)> AND <TDFT-DFID(TEMP-IND>	FLDSEMS	ERROR	RECORD
IF GROUP LEVEL IS UNIQUE KEY, AND SUBCOMPONENT IS KEY, IT MUST BE DUPLICATE KEY. SEE <DF-ID> AND <TDFT-DFID>	RECSEMS	ERROR	RECORD
ILLEGAL DEPENDENT CARDINALITY	CHKCARD	ERROR	RELATION
ILLEGAL DEPENDENT CARDINALITY	ALTCARD	ERROR	
ILLEGAL INPUT PARAMETERS FOR RETRIEVAL	ALGCHK	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
ILLEGAL INPUT PARAMETERS FOR UPDATE	ALGCHK	ERROR	CSIS
ILLEGAL JOIN OF ENTITY <IND-ECNAME> AND ENTITY <DEP-ECNAME> ON NON KEY ATTRIBUTE	VALVWRC	ERROR	VIEW
ILLEGAL OUTPUT PARAMETERS FOR RETRIEVAL	ALGCHK	ERROR	CSIS
ILLEGAL OUTPUT PARAMETERS FOR UPDATE	ALGCHK	ERROR	CSIS
IMS DATA BASE COULD NOT BE INSERTED	DEFIMS	ERROR	DATABASE
IMS DATA FIELD <DF-ID> COULD NOT BE INSERTED	DEFFLD	ERROR	DATAFIELD
IMS DATA FIELD <DF-ID> COULD NOT BE INSERTED	DEFDF	ERROR	DATAFIELD
IMS FIELD INFORMATION MISSING FOR SOME FIELD	DEFFLD	ERROR	DATAFIELD
IMS REQUIRES PSB CLAUSE	DEFIMS	ERROR	DATABASE
IMS SEGMENT INFORMATION COULD NOT BE INSERTED	DEFIMSS	ERROR	RECORD
IN OPENING FILE	TERSTR	ERROR	DATABASE
INCOMPATABLE DATA TYPES BETWEEN TAG/DF	MAPADF	ERROR	USERTYPE
INCOMPATIBLE DATA TYPES BETWEEN DI AND AUC	CHKDOMS	ERROR	DATAITEM
INCOMPATIBLE DOMAINS FOR ITEM: <DI-NAME>	CHKDOMS	ERROR	DOMAIN
INCOMPATIBLE DOMAINS FOR TAG: <TAG-NAME> INCOMPATIBLE DOMAINS FOR ITEM:<DI-NAME>	CHKDOMS	ERROR	DOMAIN
INDEPENDENT AND DEPENDENT ENTITIES MUST BE DIFFERENT	CRTREL	ERROR	ENTITY
INDEPENDENT CARDINALITY <value-x> TOO LARGE - TRUNCATED TO 99	CHKCARD	WARNING	RELATION

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
INDEX FIELD CANNOT REDEFINE OR GBE GROUP OR KEY OR COMPONENT OR REPEATING FIELD. SEE FIELD - <DF-ID>	RECSEMS	ERROR	RECORD
INDEX FIELD COULD NOT BE LOCATED	ALTFLDS	ERROR	DATAFIELD
INDEX MUST BE NUMERIC DATA TYPE FOR INDEX <TDFT-DFID (TDFT-INDEX)>	FLDSEMS	ERROR	RECORD
INDEX MUST BE NUMERIC DATA. SEE - <TDFT-DFID(TDFT-INDEX)>	FLDSEMS	ERROR	RECORD
INDEX MUST BE NUMERIC. SEE FIELD - <DF-ID>	RECSEMS	ERROR	RECORD
INHERITED ATTRIBUTE NOT DELETED FOR TAG <NEW-TAG-NO>	DELMIGRC	ERROR	ATTRIBUTE
INHERITED ATTRIBUTE NOT DELETED FOR TAG <NEW-TAG-NO>	DELMIGK	ERROR	ATTRIBUTE
INHERITED ATTRIBUTE NOT DELETED FOR TAG <NEW-TAG-NO>	DLMIGRC	ERROR	ATTRIBUTE
INHERITED ATTRIBUTE NOT DELETED FOR TAG <NEW_TAG_NO>	DMIGKRC	ERROR	TAG
INITIALIZATION FAILED	INITSES	ERROR	SYNTAX
INSERT NOT SUCCESSFUL	MAPAREC	ERROR	CSIS
INSERT OF DISTRIBUTED RULES FOR ENTITY <EC_NO> HAS FAILED	CRTMAP	ERROR	ENTITY
INTEGRATED_MODEL CANNOT BE RENAMED	RENMOBJ	ERROR	MODEL
INTEGRITY TEST FAILED DELETING <PARM-NAME>	DRPARMS	ERROR	MODULE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
INTEGRITY TEST FAILED DELETING ALL PARMS FOR <MOD-ID>	ALTSMOD	ERROR	MODULE
INTEGRITY TEST FAILED WHEN DELETING PARAMETERS FOR <MOD-ID>	DRPSMOD	ERROR	MODULE
INTERNAL TABLE OVERFLOW - RC-NAME	ECLOOK	ERROR	
INTERNAL TABLE OVERFLOW - EC-NAME	SELECXR	ERROR	VIEW
INTERNAL TABLE OVERFLOW FOR SEC-DECOMP-LIST	VALVWRC	ERROR	VIEW
INTERNAL TABLES FOR VIEW COMMAND CAN NOT BE CREATED	CRTVIEW	ERROR	VIEW
INTRA-MODEL COPY ATTRIBUTE ALIASES ARE NOT PERMITTED	ICOPATT	WARNING	ATTRIBUTE
INTRA-MODEL COPY ATTRIBUTE ALIASES NOT PERMITTED	FCOPATT	WARNING	ATTRIBUTE
INTRA-MODEL COPY CANNOT HAVE WITH <WITH-TYPE> CLAUSE	COPENT	ERROR	MODEL
INVALID ATTRIBUTE <AC-NAME> FOR THIS MODEL	DRPAC	ERROR	ATTRIBUTE
INVALID DATA BASE / HOST NAME	GETRDH	ERROR	DATABASE
INVALID DATA BASE / HOST NAME	GETDBST	ERROR	DATABASE
INVALID DOMAIN SPECIFIED <DOM-NAME>	CHGDOM	ERROR	DOMAIN
INVALID IMS SEGMENT START BYTE	DEFDF	ERROR	DATAFIELD
INVALID MODIFY <SHOW-RC>	MODAUCE	ERROR	RELATION

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
INVALID PARTITION RECORD NAME FOR THIS TAG	ADDMAP	ERROR	RECORD
INVALID RELATION CLASS NAME <TEMP-RC-NAME> DOES NOT EXIST	VERRELS	ERROR	RELATION
INVALID SEC STRUCTURE	VERRELS	ERROR	VIEW
INVALID SEGMENT SIZE	DEFIMSS	ERROR	RECORD
INVALID SYNTAX---> <command-part>	YYERROR	WARNING	SYNTAX
INVALID WHERE CLAUSE: TYPE2 MUST BE SEPERATED FROM TYPE3	STRQCRF	ERROR	VIEW
ITEMS MISSING FROM PARAMETER LIST	VEROBJ1	ERROR	OBJECT
ITEMS MISSING FROM PARAMETER LIST	VEROBJ	ERROR	OBJECT
KEY CLASS <KC-NAME> ALREADY EXISTS FOR THIS ENTITY	ADDKC	WARNING	KEY
KEY CLASS <KC-NAME> CANNOT BE DELETED	DRPKC	ERROR	KEY
KEY CLASS <KC-NAME> NOT FOUND IN THIS ENTITY	DRPKC	ERROR	KEY
KEY CLASS <KC-NAME> NOT MIGRATED BY THIS RELATION CLASS	DRPMIG	ERROR	KEY
KEY CLASS <kc-name> IS INVALID	ADDMIG	ERROR	KEY
KEY CLASS CAN NOT BE CREATE FOR COPIED ENTTIY	TERKEY	ERROR	KEY

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
KEY CLASS MEMBER NOT DELETED FOR ATTRIBUTE CLASS <attribute class number>	DELAUC	ERROR	KEY
KEY CLASS MEMBER NOT DELETED FOR KEY CLASS <key class number>	DELAUCK	ERROR	KEY
KEY CLASS MEMBER NOT DELETED FOR TAG <NEW-TAG-NO>	DLMIGRC	ERROR	KEY
KEY CLASS MEMBER NOT DELETED FOR TAG <NEW-TAG-NO>	DMIGKRC	ERROR	TAG
KEY CLASS NAME <KC-NAME> ALREADY EXISTS	ALTENTK	ERROR	KEY
KEY CLASS NAME <KC-NAME> DOES NOT EXIST	ALTENTK	ERROR	KEY
KEY CLASS NOT DELETED FOR KEY CLASS <key class number>	DELMTKC	ERROR	KEY
KEY-CLASS <KC-NAME> NOT FOUND ON INDEPENDENT ENTITY	DRPMIG	ERROR	KEY

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
KEY-PROCESSED TABLE OVERFLOW	AOAKC	ERROR	KEY
KEY CLASS MEMBER NOT DELETED FOR TAG<NEW-TAG-NO>	DELMIGRC	ERROR	KEY
KEYWORD <KEY-WORD> NOT DELETED FROM DESC-TEXT	DRPKWC	ERROR	KEYWORD
KEYWORD <KEY-WORD> TO BE DROPPED DOES NOT EXIST	DRPKW	ERROR	KEYWORD
KEYWORD <OBJ-ID-1> DOES NOT EXIST	VEROBJ	ERROR	KEYWORD
KEYWORD <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	KEYWORD
KEYWORD ATTRIBUTE ASSOCIATION NOT DELETED FOR <key-word>	DRPKWC	ERROR	KEYWORD
KEYWORD COULD NOT BE DELETED <key-word>	DRPKWC	ERROR	KEYWORD
KEYWORD DOES NOT EXIST <key-word>	DRPKWC	ERROR	KEYWORD
KEYWORD NOT ADDED FOR <attribute class name>	ALTATT	ERROR	KEYWORD
KEYWORD NOT DROPPED FOR <attribute class name>	ALTATT	ERROR	KEYWORD
KEYWORD-ENTITY ASSOCIATION NOT DELETED FOR <key-word>	DRPKWC	ERROR	KEYWORD
KEYWORD-RELATION ASSOCIATION NOT DELETED FOR <key-word>	DRPKWC	ERROR	KEYWORD
KEYWORDS CAN NOT BE DROPPED FOR ENTITY <ec-name>	DRPENT	ERROR	ENTITY

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
KEYWORDS CANNOT BE DROPPED	DRPKW	ERROR	KEYWORD
KEY_CLASS_MEMBER NOT DELETED FOR TAG-NO <tag-number>	DELMIGK	ERROR	KEY
LEFT DEPENDENT CARDINALITY <VALUE-X> TOO LARGE - VALUE UNCHANGED	ALTCARD	WARNING	
LEFT DEPENDENT CARDINALITY <value-x> TOO LARGE - TRUNCATED TO ZERO	CHKCARD	WARNING	RELATION
LESS THAN 2 ENTITIES IN RECORD UNION DEFINITON	ALTUNIN	ERROR	CSIS
LEVEL EXCEEDED 99 LEVEL LIMIT - NO FURTHER LEVELS PROCESSED	ADDFLDS	ERROR	RECORD
LEVEL EXCEEDED 99 LEVEL LIMIT. NO FURTHER LEVELS PROCESSED	DEFFLDS	ERROR	RECORD
LINKAGE DATA FIELD <DF-ID> IS NOT DEFINED	DEFSET	ERROR	DATAFIELD
LINKAGE DATA FIELD: <DF-ID> IS NOT DEFINED	DEFSET	ERROR	DATAFIELD
LINKAGE DATA FIELD: <DF-ID> WAS INSERTED	DEFSET	ERROR	DATAFIELD
LINKAGE DATA FIELD: <DF-ID> WAS NOT INSERTED	DEFSET	ERROR	DATAFIELD
LINKAGE DATA FIELD:<DF-ID> MUST BE KNOWN TO THE DBMS	DEFSET	ERROR	DATAFIELD
LINKAGE WAS NOT FOUND FOR <DF-ID>	DEFSET	ERROR	DATAFIELD

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
LINKAGE WAS NOT FOUND FOR <DF-ID>	DEFSET	ERROR	DATAFIELD
MAP TYPE IS COMPLEX, COMPLEX MAPPINGS MUST BE DROPPED	DRPMAP	ERROR	CSIS
MAPPING EXISTS FOR RELATION <RC-NAME>	DRPRCE	ERROR	RELATION
MAPPING NOT ALLOWED TO A REPEATING, GROUP DATA FIELD	MAPADF	ERROR	DATAFIELD
MAXIMUM TABLE SIZE OCCURED AT DATA FIELD <DF-ID> NO FURTHER ENTRIES PROCESSED	ADDFLDS	ERROR	RECORD
MAXIMUM TABLE SIZE OCCURED AT DATA FIELD <DF-ID>. NO FURTHER ENTRIES WILL BE PROCESSED	DEFFLDS	ERROR	RECORD
MEMBER FROM RECORD-SET NOT DELETED	SELSTM	ERROR	RECORD
MEMBER NOT DELETED FROM SET-TYPE-MEMBER	SELSTM	ERROR	RECORD
MEMBER RECORD: <MEMBER-ID> DOES NOT EXIST	DEFSET	ERROR	DATAFIELD
MEMBER RECORD: <MEMBER-ID> DOES NOT EXIST	DEFSET	ERROR	DATAFIELD
MIGRATED KEY CLASSES NOT DELETED FOR ENTITY <EC-NAME>	DRPENT	ERROR	ENTITY
MIGRATING KEYS NOT DELETED FOR KEY CLASS <key class number>	DELAUCK	ERROR	KEY

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
MISSING OR INCOMPLETE RELATION BETWEEN <IND-ECNAME> AND <DEP-ECNAME>	VALVWRC	ERROR	VIEW
MOD TABLE OVERFLOW ERROR	GENCMPX	ERROR	CSIS
MOD-TABLE OVERFLOW ERROR	GENCMPX	ERROR	
MODEL 1 <MOD-NAME> DOES NOT EXIST	MRGMOD	ERROR	MODEL
MODEL 1 NOT COPIED INTO MODEL 3	MRGMOD	ERROR	MODEL
MODEL 2 <MOD-NAME> DOES NOT EXIST	MRGMOD	ERROR	MODEL
MODEL <MOD-NAME2> CANNOT BE MERGED	MRGMOD2	ERROR	MODEL
MODEL <MOD-NAME2> CANNOT BE MERGED	MRGNODE	ERROR	MODEL
MODEL <MOD-NAME> ALREADY EXISTS	CPYMOD	ERROR	MODEL
MODEL <MOD-NAME> DOES NOT EXIST	CPYMOD	ERROR	MODEL
MODEL <MODEL-NAME> COULD NOT BE MARKED AS CHECKED	CHKMOD	ERROR	MODEL

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
MODEL <MODEL-NAME> DOES NOT EXIST	CHKMOD	ERROR	MODEL
MODEL <MODEL-NAME> DOES NOT EXIST	CPYDES	ERROR	MODEL
MODEL <MODEL-NAME> DOES NOT EXIST	DRPMOD	ERROR	MODEL
MODEL <MODEL-NAME> HAS BEEN DROPPED	DRPMOD	WARNING	MODEL
MODEL <MODEL-NAME> HAS NO BOTTOM ENTITY	CHKLOOP	WARNING	MODEL
MODEL <MODEL-NAME> HAS NO TOP ENTITY	CHKLOOP	WARNING	MODEL
MODEL <MODEL-NAME> NOT DELETED FROM DESC-TEXT	DRPMOD	ERROR	MODEL
MODEL <MODEL-NAME> NOT DELETED FROM MODEL-CLASS	DRPMOD	ERROR	MODEL
MODEL <MODEL-NAME> WHERE ATTRIBUTE IS TO BE COPIED DOES NOT EXIST	COPATT	ERROR	ATTRIBUTE
MODEL <OBJ-ID-1> DOES NOT EXIST	VEROBJ	ERROR	MODEL
MODEL <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	MODEL
MODEL <model-name1> DOES NOT EXIST	CMPMOD	ERROR	MODEL
MODEL <model-name2> DOES NOT EXIST	CMPMOD	ERROR	MODEL
MODEL <model-name> ALREADY EXISTS	CRTMOD	ERROR	MODEL

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
MODEL <model-name> CAN NOT BE ALTERED	ALTMOD	ERROR	MODEL
MODEL <model-name> CAN NOT BE CREATED	CRTMOD	ERROR	MODEL
MODEL <model-name> HAS BEEN ALTERED	ALTMOD	WARNING	MODEL
MODEL <model-name> HAS BEEN CREATED	CRTMOD	WARNING	MODEL
MODEL <model-name> HAS MORE THAN 400 RELATIONS	ADDRCEC	ERROR	MODEL
MODEL <model-name> TO BE ALTERED DOES NOT EXIST	ALTMOD	ERROR	MODEL
MODEL TYPE <DB-MODEL> ALREADY DEFINED FOR DBMS <DBMS-NAME>	ALTDBMS	WARNING	DATABASE
MODEL TYPE MUST BE R, H, N, F OR S	ALTDBMS	ERROR	DATABASE
MODEL TYPE MUST BE R, H, OR N	DEFDBMS	ERROR	DATABASE
MODULE HAS BEEN PRECOMPILED AGAINST DATABASE	DRPDB	ERROR	DATABASE
MODULE INSTANCE/PREFERENCE # SHOULD BE PIC 99	DRPALG	ERROR	CSIS/CSES
MODULE NOT A COMPLEX MAPPING ALGORITHM-CAN'T ALTER	ALTSMOD	ERROR	MODULE
MODULES MUST BE DROPPED PRIOR TO DROPPING VIEW	DRPVIEW	WARNING	VIEW
MORE RELATIONS EXIST WITH INDEPENDANT ENTITY: <IND-EC> AND DEPENDANT ENTITY <DEP-EC>	FRTOREL	WARNING	RELATION

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
MULTIPLE MAPPINGS NOT PERMITTED FOR THIS PREF: <DBNAME.RECNAME> IS NOT HP	ADDMAP	ERROR	CSIS
MULTIPLE MEMBERS ARE NOT ALLOWED IN: <DBMS-NAME>	DEFSET	ERROR	SET
MUST BE AT LEAST 2 ENTITIES TO CREATE A UNION	ADDUNIN	ERROR	CSIS
NAMES CAN NOT BE DROPPED FOR ENTITY <ec-name>	DRPENT	ERROR	ENTITY
NDML COMMIT NOT PERFORMED	COMMIT	ERROR	
NDML ERROR - ALTER ALTER NOT PERFORMED	ALTSMAP	ERROR	CSIS
NDML ERROR - COPY HOST 'ALL' NOT COMPLETED	CPYHST	ERROR	
NDML ERROR - COPY HOST (HOST-ID) NOT COMPLETED	CPYHST	ERROR	
NDML ERROR - CS-IS MAPPING NOT DROPPED	DELMTAG	ERROR	CSIS
NDML ERROR CHANGING A CS-IS MAPPING PREFERENCE	ALTMAP	ERROR	CSIS
NDML ERROR CS-IS MAPPING NOT DROPPED	DRPMAP	ERROR	CSIS/CSES
NDML ERROR CS-IS MAPPING NOT DROPPED	DRPSPRF	ERROR	CSIS
NDML ERROR DELETING A CS-IS AUC SET MAP	DRPSMAP	ERROR	CSIS
NDML ERROR DELETING A CS-IS DF MAP	DRPSMAP	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NDML ERROR DELETING A DF-MAPPING	DRPDFMP	ERROR	CSIS
NDML ERROR DELETING A RC SET MAPPING	DRPSMAP	ERROR	CSIS
NDML ERROR DELETING A SET MAPPING	DRPSTMP	ERROR	CSIS
NDML ERROR DELETING ALL PARAMETERS FOR <MOD-ID>	ALTSMOD	ERROR	MODULE
NDML ERROR DELETING DISTRIBUTED RULES	SELURT	ERROR	CSIS
NDML ERROR DELETING EC_RT_MAPPING	SELURT	ERROR	CSIS
NDML ERROR DELETING ENTITY <EC-NAME> FROM RECORD UNION	ALTUNIN	ERROR	CSIS
NDML ERROR DELETING FROM DISTRIBUTED RULES	ALTUNIN	ERROR	CSIS
NDML ERROR DELETING FROM DISTRUBTED_RULES	DRPALG	ERROR	CSIS/CSES
NDML ERROR DELETING FROM EC_RT_MAPPING	ALTUNIN	ERROR	CSIS
NDML ERROR DELETING FROM EC_RT_MAPPING	DRPALG	ERROR	CSIS/CSES
NDML ERROR DELETING FROM EC_RT_MAPPING	DRPECRT	ERROR	CSIS
NDML ERROR DELETING MOD. PARM. <PARM-NAME>	DRPARMS	ERROR	MODULE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NDML ERROR DELETING RECORD UNION	SELURT	ERROR	CSIS
NDML ERROR DELETING SOFTWARE MODULE	DRPSMOD	ERROR	MODULE
NDML ERROR DELETING TEXT <DATA-TYPE-NAME>	DRPDT	ERROR	USERTYPE
NDML ERROR GETTING DAT-TYP <DATA-TYPE-NAME>	DRPDT	ERROR	USERTYPE
NDML ERROR IN VERIFYING DB APPLICATIONS	DELMTAG	ERROR	CSIS
NDML ERROR IN VERIFYING TYPE	PROCDT	ERROR	NDML
ERROR INSERTING CS-IS DF MAP	MAPADF	ERROR	CSIS
NDML ERROR INSERTING CS-IS SET MAP VALUES	MAPASET	ERROR	CSIS
NDML ERROR INSERTING DF VALUES FROM STRUCTURE	MAPADF	ERROR	CSIS
NDML ERROR INSERTING MODULE PARAMETER	DEFSMOD	ERROR	MODULE
NDML ERROR INSERTING RANGE: <DOM-BEG-VALUE> THRU <DOM-END-VALUE>	CRTVMOD	ERROR	MODULE
NDML ERROR INSERTING RECORD UNION	ADDUNIN	ERROR	CSIS
NDML ERROR INSERTING SET VALUES FROM STRUCTURE	MAPASET	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NDML ERROR INSERTING SOFTWARE MODULE	DEFSMOD	ERROR	MODULE
NDML ERROR INSERTING VALUE: <DOM-VALUE>	CRTVMOD	ERROR	DOMAIN
NDML ERROR ON SELECT OF DOMAIN_RANGE	ALTVMOD	ERROR	DOMAIN
NDML ERROR ON SELECT OF DOMAIN_VALUE	ALTVMOD	ERROR	DOMAIN
NDML ERROR RETRIEVING DF_SET_LINKAGE	GENSET	ERROR	
NDML ERROR RETRIEVING DOMAIN RANGES	GENDOM	ERROR	DOMAIN
NDML ERROR RETRIEVING DOMAIN VALUES	GENDOM	ERROR	DOMAIN
NDML ERROR RETRIEVING EC_NO FROM ENTITY_NAME	ADDUNIN	ERROR	CSIS
NDML ERROR RETRIEVING RANGE	ALTVLRG	ERROR	DOMAIN
NDML ERROR RETRIEVING RANGES	ALTVLRG	ERROR	DOMAIN
NDML ERROR RETRIEVING STD. DATA TYPE	GENDOM	ERROR	USERTYPE
NDML ERROR RETRIEVING USER DATA TYPES	GENDOM	ERROR	USERTYPE
NDML ERROR RETRIEVING USER DATA TYPES	GENDOM	ERROR	USERTYPE
NDML ERROR RETRIEVING VALUE <DOM-VALUE>	ALTVLRG	ERROR	DOMAIN

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NDML ERROR RETRIEVING VALUES	ALTVLRG	ERROR	DOMAIN
NDML ERROR RETRIEVING VERIF. MOD. FOR DOMAIN	GENDOM	ERROR	DOMAIN
NDML ERROR SELECTING DATA ITEM LIST	GENVIEW	ERROR	VIEW
NDML ERROR SELECTING ENTITY/TAG NAME LIST	GENVIEW	ERROR	VIEW
NDML ERROR SELECTING FOR ENTITY EC_RT_MAPPING	DRPALG	ERROR	CSIS/CSES
NDML ERROR SELECTING FROM EC_RT_MAPPING FOR <EC-NAME>	ALTUNIN	ERROR	CSIS
NDML ERROR SELECTING FROM ENTITY_NAME FOR <EC-NAME>	ALTUNIN	ERROR	CSIS
NDML ERROR SELECTING FROM RP-SUBROUTINE	CHKRPS	ERROR	DATABASE
NDML ERROR SELECTING PARAMETERS INTO STRUCTURE	ALTSMOD	ERROR	MODULE
NDML ERROR SELECTING RECORD UNIONS	ALTUNIN	ERROR	CSIS
NDML ERROR SELECTING SET	CPYSET	ERROR	
NDML ERROR SELECTING SET MEMBERS	GENSET	ERROR	
NDML ERROR SELECTING STD.DATA TYPE	GENDOM	ERROR	USERTYPE
NDML ERROR UPDATING SOFTWARE MODULE	ALTSMOD	ERROR	MODULE
NDML ERROR VERIFYING DB APPLICATIONS	DRPMAP	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NDML ERROR VERIFYING ENTITY <EC-NAME> IN RECORD UNION	ALTUNIN	ERROR	CSIS
NDML ERROR VERIFYING MODULE PARAMETER	DRPARMS	ERROR	MODULE
NDML ERROR VERIFYING PARAMETER	DEFSMOD	ERROR	MODULE
NDML ERROR VERIFYING SOFTWARE MOD.	DEFSMOD	ERROR	MODULE
NDML ERROR VERIFYING SOFTWARE MOD.	DRPSMOD	ERROR	MODULE
NDML ERROR VERIFYING SOFTWARE MOD.	ALTSMOD	ERROR	MODULE
NDML ERROR WHEN DELETING FROM MODULE_PARAMETER TABLE	DRPDOM	ERROR	MODULE
NDML ERROR WHEN DELETING FROM MODULE_PARAMTER TABLE	ALTVMOD	ERROR	MODULE
NDML ERROR WHEN DELETING FROM SOFTWARE_MODULE TABLE	ALTVMOD	ERROR	MODULE
NDML ERROR WHEN DELETING FROM SOFTWARE_MODULE TABLE	DRPDOM	ERROR	MODULE
NDML ERROR WHEN DELETING FROM VERIF-MODULE TABLE	DRPDOM	ERROR	DOMAIN
NDML ERROR WHEN DELETING FROM VERIF-MODULE TABLE	ALTVMOD	ERROR	
NDML ERROR WHEN DROPPING ALL RANGES	DRPDOM	ERROR	DOMAIN
NDML ERROR WHEN DROPPING ALL RANGES	ALTVLRG	ERROR	

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NDML ERROR WHEN DROPPING ALL VALUES	DRPDOM	ERROR	DOMAIN
NDML ERROR WHEN DROPPING ALL VALUES	ALTVLRG	ERROR	DOMAIN
NDML ERROR WHEN DROPPING RANGE	ALTVLRG	ERROR	DOMAIN
NDML ERROR WHEN DROPPING VALUE	ALTVLRG	ERROR	DOMAIN
NDML ERROR WHEN INSERTING INTO SOFTWARE_MODULE TABLE	ALTVMOD	ERROR	MODULE
NDML ERROR WHEN INSERTING INTO MODULE_PARAMETER <CRT-APNAME>	CRTVMOD	ERROR	MODULE
NDML ERROR WHEN INSERTING INTO MODULE_PARAMETER TABLE	ALTVMOD	ERROR	MODULE
NDML ERROR WHEN INSERTING INTO SOFTWARE_MODULE <CRT-APNAME>	CRTVMOD	ERROR	MODULE
NDML ERROR WHEN RETRIEVING AP-NAME FROM VERIF-MODULE TABLE	DRPDOM	ERROR	DOMAIN
NDML ERROR WHEN RETRIEVING AP-NAME FROM VERIF-MODULE TABLE	CRTMACR	ERROR	
NDML ERROR WHEN RETRIEVING AP-NAME FROM VERIF-MODULE TABLE	ALTVMOD	ERROR	
NDML ERROR WHEN RETRIEVING DATA TYPE: <DATA-TYPE-NAME>	ALTDT	ERROR	USERTYPE
NDML ERROR WHEN RETRIEVING STND. DATA TYPE	ALTVLRG	ERROR	DOMAIN

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NDML ERROR WHEN TRYING TO UPDATE VERIF-MODULE <CRT-APNAME>	CRTVMOD	ERROR	DOMAIN
NDML ERROR WHEN TRYING TO UPDATE VERIF_MODULE	ALTVMOD	ERROR	MODULE
NDML ERROR WHEN VERIFYING OR DELETING UNION	DRPUNIN	ERROR	CSIS
NDML ERROR WHILE SELECTING EC_RT_MAPPING	SELURT	ERROR	CSIS
NDML ERROR WHILE SELECTING RECORD UNION	ADDUNIN	ERROR	CSIS
NDML ERROR- COPY MODULE 'ALL' NOT COMPLETED	CPYMODU	ERROR	MODULE
NDML MODIFY FAILED	ALTMAP	ERROR	CSIS
NDML MODIFY NOT SUCCESSFUL	ALTMAP	ERROR	CSIS
NDML ROLLBACK UNSUCCESSFUL	ROLBACK	ERROR	
NDML SELECT FAILED FOR ENTITY <EC-NO>	DRPSMAP	ERROR	ENTITY
NDML SELECT FAILED FOR ENTITY <EC_NO>	MAPASET	ERROR	ENTITY
NDML SELECT FAILED FOR ENTITY <EC_NO>	MAPADF	ERROR	ENTITY
NDML SELECT HAS FAILED	ALTMAP	ERROR	CSIS
NEW ATTRIBUTE <TAG-NAME> COULD NOT BE ADDED AS OWNED	TERATTR	ERROR	ATTRIBUTE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NEW ATTRIBUTE <TAG-NAME> IS ALREADY OWNED	TERATTR	ERROR	ATTRIBUTE
NEW DESCRIPTION FOR ENTITY <TO-EC-NAME> NOT CREATED	TRADIR	ERROR	ENTITY
NEW OBJECT NAME ALREADY EXISTS, ID IS : <NEW-ID>	RENMOBJ	ERROR	OBJECT
NEXT SET-NO NOT AVAILABLE	DEFSET	ERROR	SET
NO CLAUSES APPLY TO FILLER - ONLY FILLER SIZE IS NEEDED	RECSEMS	ERROR	RECORD
NO DESCRIPTIONS EXIST IN THE CDM FOR <OBJECT-TYPE>	CPDDRAL	WARNING	DESCRIPTION
NO DISTRIBUTED RULES EXIST FOR ENTITY <EC_NO>	CRTMAP	ERROR	ENTITY
NO EC_RT MAPPING EXIST FOR ENTITY <EC-NAME> AND RECORD <RT-ID>	ALGCHK	ERROR	CSIS
NO EC_RT_MAPPING FOR <EC-NAME> <RT-ID>	ADDUNIN	ERROR	CSIS
NO ENTITY NAME ASSIGNED ABBREVIATION <EC-ABBR-NAME>	BLVWLST	ERROR	VIEW
NO ENTITY TO RECORD MAP FOR ENTITY <EC_NO> & RT-NO <RT_NO>	MAPADF	ERROR	ENTITY
NO ENTITY TO RECORD MAP FOR ENTITY<EC_NO> & RT_NO<RT_NO>	MAPASET	ERROR	ENTITY

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NO ENTITY TO RECORD MAPPING EXISTS FOR ENTITY <EC-NO> CREATE MAP COMMMAND SHOULD BE USED	ALTMAP	ERROR	CSIS
NO MAP EXISTS FOR THIS RELATION CLASS	DRPSMAP	ERROR	CSIS
NO MAP FOUND FOR SPECIFIED ENTITY AND TAG	DRPPRF1	ERROR	CSIS
NO MAPPING EXISTS FOR <DBNAME> <RT-ID>	SELMREC	WARNING	DATABASE
NO MAPPING EXISTS FOR ENTITY < >	SELMTAG	ERROR	ENTITY
NO MAPPING EXISTS FOR THIS RECORD TYPE	DRPSMAP	ERROR	CSIS
NO MORE TAG NUMBERS AVAILABLE	ADDMIG	ERROR	TAG
NO PARAMETERS FOR MODULE <MOD-ID> CAN'T ADD BEFORE/AFTER	ALTSMOD	ERROR	MODULE
NO PRIVELEDGE TO DROP INTEGRATESD_MODEL	DRPMOD	ERROR	MODEL
NO RANGES FOUND TO BE DROPPED	ALTVLRG	WARNING	DOMAIN
NO TAGS EXIST-NONE WERE DROPPED	DRPALTG	WARNING	CSIS
NO VALUES FOUND TO DROP	ALTVLRG	WARNING	DOMAIN
NO. OF DATA ITEMS MUST EQUAL NO. OF TAGS SELECTED	PNOFROM	ERROR	SYNTAX
NON-NUMERIC DISCRIMINATOR VALUE FOR NUMERIC DATAFIELD	ADDUNIN	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
NOT ENOUGH JOIN CONDITIONS SPECIFIED FOR VIEW	VALVWRC	ERROR	VIEW
NUMBER OF DATA ITEMS MUST EQUAL NUMBER OF TAG NAMES	REMVIEW	ERROR	VIEW
OBJECT INVALID FOR RENAMING	RENMOBJ	ERROR	OBJECT
OBJECT NOT RENAMED FOR : <OLD-OBJ-NO>	RENMOBJ	ERROR	OBJECT
OBJECT TO BE RENAMED DOES NOT EXIST, ID IS :<OLD-ID>	RENMOBJ	ERROR	OBJECT
OCCURS DEPENDING ON FIELD <FIELD-NAME-3> IS NOT DEFINED	ADDFLDS	ERROR	RECORD
OCCURS MUST BE GREATER THAN 1 WHEN DEPENDING ON IS PRESENT. SEE FIELD <DF-ID>	RECSEMS	ERROR	RECORD
OCCURS SHOULD BE GREATER THAN 1 SEE FIELD - <TDFT-DFID (TDFT-INDEX)>	FLDSEMS	ERROR	RECORD
OCCURS SHOULD BE GREATER THAN 1 WHEN DEPENDING ON IS PRESENT. SEE FIELD - <TDFT-DFID<TDFT-INDEX)>	FLDSEMS	ERROR	RECORD
OCCURS SHOULD BE GREATER THAN 1 SEE FIELD - <DF-ID>	RECSEMS	ERROR	RECORD
OISCR ERROR IN PROCMD - RCODE IS <RCODE>	PROCMD	ERROR	
OLD STD COULD NOT BE UPDATED TO USER FOR: <DTN-OLDSTD>	ALTDT	ERROR	USERTYPE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
OLD TAG NAME DOES NOT EXIST IN ENTITY	ALTTAG	ERROR	TAG
OLD-TAG-MIGRATION-TBL OVERFLOWED	OLDTAGM	ERROR	TAG
OPNFRM ERROR IN INITSES - RCODE IS <RCODE>	INITSES	ERROR	
ORACLE DATA BASE COULD NOT BE INSERTED	DEFORCL	ERROR	DATABASE
ORACLE UPDATE ERROR <SHOW-RC>	MODIAUC	ERROR	TAG
OUTPUT TO FILE OR SCREEN HAS NOT BEEN ESTABLISHED	CPYMOD	ERROR	
OWNED ATTRIBUTE CAN NOT BE CREATED	FNDOAC	ERROR	ATTRIBUTE
OWNER AND SET MEMBERS NOT DELETED.	DRPREC	ERROR	RECORD
OWNER FROM RECORD-SET NOT DELETED	SELRSET	ERROR	RECORD
OWNER NOT DELETED FROM SET-TYPE-MEMBER	SELRSET	ERROR	RECORD
OWNER RECORD: <OWNER-ID> DOES NOT EXIST	DEFSET	ERROR	RECORD
OWNER RECORD:<OWNER-ID> DOES NOT EXIST	DEFSET	ERROR	SET
OWNERSHIP NOT ALTERED FOR <AC-NAME>	ALTATT	ERROR	ATTRIBUTE
PARAMETER <PARM-NAME> ALREADY EXISTS	ADDPRMS	ERROR	MODULE
PARAMETER EXISTS FOR MODULE	DEFSMOD	ERROR	MODULE

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
PARM NAME DOESN'T EXIST FOR <MOD-ID>	DRPARMS	ERROR	MODULE
PARTITION COULD NOT BE INSERTED IN CDM DATA BASE	CRTPART	ERROR	ENTITY
PARTITION COULD NOT BE SUCCESSFULLY INSERTED	ALTPART	ERROR	ENTITY
PARTITION NUMBER ALREADY EXISTS	CRTPART	ERROR	ENTITY
PARTITION NUMBER DOES NOT EXIST ON CDM	HPNLOOK	ERROR	ENTITY
PARTITION RECORD COULD NOT BE DROPPED.	VALDROP	ERROR	ENTITY
PASSIVE MAPPINGS MUST HAVE PREF NUMBERS BETWEEN 51 AND 99	CRTMAP	ERROR	CSIS
PASSIVE MAPPINGS MUST HAVE PREF NUMBERS BETWEEN 51 AND 99	ALTMAP	ERROR	CSIS
PASSWORD CLAUSE REQUIRED FOR ORACLE DBMS	DEFORCL	ERROR	DATABASE
PDATA ERROR ON CMDLINE0 IN PROCMD - RCODE IS <RCODE>	PROCMD	ERROR	
PDATA ERROR ON CMDLINE1 IN PROCMD - RCODE IS <RCODE>	PROCMD	ERROR	
PDATA ERROR ON CMDLINE2 IN PROCMD - RCODE IS <RCODE>	PROCMD	ERROR	

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
PDATA ERROR ON CURCOM IN PROCMD- RCODE IS <RCODE>	PROCMD	ERROR	
PDATA ERROR ON CURDB IN PROCMD - RCODE IS <RCODE>	PROCMD	ERROR	
PDATA ERROR ON CURMOD IN PROCMD - RCODE IS <RCODE>	PROCMD	ERROR	
PDATA ERROR ON CUROPT IN PROCMD- RCODE IS <RCODE>	PROCMD	ERROR	
POSITIONAL PARM <POS-PARM-NAME> NOT FOUND	PROCPRM	ERROR	MODULE
PREFERENCE NUMBER <IN-PREF-NO> IS OUT OF RANGE <1-99>	DRPMAP	ERROR	CSIS
PREFERENCE NUMBER <PREF-NO> IS OUT OF RANGE <1-99>	CRTMAP	ERROR	ENTITY
PREFERENCE NUMBER <PREF-NO> IS OUT OF RANGE <1-99>	ALTMAP	ERROR	CSIS
PREVIOUS MAPPING CONDITIONS FOR SET <SET> WERE NOT SATISFIED	MAPASET	ERROR	CSIS
PREVIOUS TAG MAPPING DOES NOT CONFORM WITH THE ALGORITHM	ALGCHK	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
PRIMARY MAPPING MISSING FOR THIS TAG	ADDMAP	ERROR	CSIS
PROJECT DATA ITEM COULD NOT BE DELETED FOR : <VIEW-NAME>	DRPVIEW	ERROR	VIEW
PSB <PSB-NAME> NOT DEFINED	DEFIMS	ERROR	DATABASE
RANGE COULD NOT BE INSERTED RANGE = <BEG-VALUE> THRU <END-VALUE>	ALTVLRG	ERROR	DOMAIN
RANGE INCORRECT FOR BEG. VALUE - END. VALUE: <DOM-RANGE>	CRTVLRG	ERROR	DOMAIN
RANGE TO DROPPED NOT FOUND: <BEG-VALUE> THRU <END-VALUE>	ALTVLRG	WARNING	DOMAIN
RC KEYWORD ALREADY EXISTS	ADDKWR	ERROR	KEYWORD
RCNO-LIST OVERFLOWED, MORE RELATIONS EXIST	FRTOREL	WARNING	RELATION
RECORD <REC-NAME> DOES NOT EXIST	DRPSMAP	ERROR	RECORD
RECORD <REC-NAME> DOES NOT EXIST	MAPAREC	ERROR	CSIS
RECORD <RT-ID-OF-OWNER> DOES NOT EXIST	ALTSMAP	ERROR	RECORD
RECORD <RT-ID> ALREADY EXISTS	DEFREC	ERROR	RECORD
RECORD <RT-ID> COULD NOT BE INSERTED	DEFREC	ERROR	RECORD
RECORD <RT-ID> DOES NOT EXIST	RETRFLD	ERROR	RECORD
RECORD <RT-ID> DOES NOT EXIST	VEROBJ	ERROR	RECORD
RECORD <RT-ID> DOES NOT EXIST	ALTFLDS	ERROR	DATAFIELD

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
RECORD <RT-ID> DOES NOT EXIST	CPYMREC	ERROR	RECORD
RECORD <RT-ID> HAS A UNION MAPPING	DRPREC	ERROR	CSIS
RECORD <RT-ID> HAS BEEN MAPPED TO A TAG	DRPREC	ERROR	CSIS
RECORD <RT-ID> IS PART OF A HORIZONTAL PARTITION MAPPING	DRPREC	ERROR	CSIS
RECORD <RT_ID> DOES NOT EXIST	VEROBJ1	ERROR	RECORD
RECORD CAN'T BE DROPPED IF FIELD IN RECORD IS STILL MAPPED	ALT1ERT	ERROR	CSIS
RECORD DOES NOT EXIST	ALTREC	ERROR	RECORD
RECORD NAME <RT-ID> DOES NOT EXIST	CPYREC	ERROR	RECORD
RECORD NAME NEEDED FOR MULTIMEMBER SET	DRPSMAP	ERROR	RECORD
RECORD NUMBER FOR OWNER RECORD <OWNER-REC-NAME> COULD NOT BE FOUND	MAPASET	ERROR	RECORD
RECORD TYPE <RT-ID> DOES NOT EXIST FOR PARTITION	VALDROP	ERROR	ENTITY
RECORD TYPE NOT FOUND.	GETDRT	ERROR	RECORD
RECORD TYPE: <RT-ID> DOES NOT EXIST	DRPREC	ERROR	RECORD
RECORD UNION CURRENTLY EXISTS - CAN'T ADD	ADDUNIN	ERROR	CSIS

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
RECORD UNION DEFINITION COMPLETELY DELETED BY	ALTUNIN	ERROR	CSIS
ALTER COMMAND THE DROP UNION COMMAND SHOULD BE USED TO ACOOMPLISH THIS			
RECORD UNION DOESN'T EXIST -CAN'T BE DROPPED	DRPUNIN	ERROR	CSIS
REDEFINED FIELD <DF-ID> IS NOT DEFINED	DEFFLDS	ERROR	RECORD
REDEFINED FIELD <FIELD-NAME-3> DOES NOT EXIST	ALTFLDS	ERROR	DATAFIELD
REDEFINED FIELD <FIELD-NAME-4> IS NOT DEFINED	ADDFLDS	ERROR	RECORD
REDEFINES FIELD AND REDEFINING FIELD MUST BE SAME TYPE KEY. SEE FIELDS <DF-ID> AND <TDFT-DFID>	RECSEMS	ERROR	RECORD
REDEFINITIONS OF TOTAL FIELDS MUST BE SPECIFIED AS A SINGLE FIELD. SEE FIELD - <DF-ID>	RECSEMS	ERROR	RECORD

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
REDEFINITIONS OF TOTAL FIELDS MUST BE SPECIFIED ED AS A SINGLE FIELD. SEE FIELD - <TDFT-DFID(TDFT-INDEX)>	FLDSEMS	ERROR	RECORD
REL CLASS <RC-NAME> DOES NOT EXIST	VEROBJ	ERROR	RELATION
REL CLASS <RC_NAME> DOES NOT EXIST	VEROBJ1	ERROR	RELATION
RELATION <IND-EC-NAME1> <RC-NAME1> <DEP-EC-NAME1> AND RELATION <IND-EC-NAME2> <RC-NAME2> HAVE MATCHING KEYWORD <KWORD-TEMP>	RRCKW2	WARNING	RELATION
RELATION CLASS <RC-NAME> ALREADY EXISTS	CRTREL	ERROR	RELATION
RELATION CLASS <RC-NAME> COULD NOT BE DROPPED	DRPREL	ERROR	RELATION
RELATION CLASS <RC-NAME> DOES NOT EXIST	ALTREL	ERROR	RELATION
RELATION CLASS <RC-NAME> DOES NOT EXIST	DRPREL	ERROR	RELATION
RELATION CLASS <RC-NAME> IS NOT MAPPED TO A SET	DRPMAP	ERROR	CSIS
RELATION CLASS <relation class name> DOES NOT EXIST	CRTMAP	ERROR	RELATION
RELATION CLASS <relation class name> DOES NOT EXIST	DRPMAP	ERROR	RELATION

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
RELATION CLASS <relation class name> DOES NOT EXIST	ALTMAP	ERROR	RELATION
RELATION CLASS CANNOT BE DELETED	DRPRCE	ERROR	RELATION
RELATION CLASS KEYWORD CANNOT BE DELETED	DRPRCE	ERROR	RELATION
RELATION DOES NOT EXIST FROM TARGET ENTITY TO THE EXCEPTED DEPENDENT ENTITY <DF-NAME>	TERREL	WARNING	RELATION
RELATION HAS MIGRATION	ADDMIG	ERROR	RELATION
RELATION HAS NO MIGRATION	DRPREL	WARNING	RELATION
RELATION KEYWORD CANNOT BE DELETED	DRPRCE	ERROR	KEYWORD
RELATION TO SET MAPPING EXISTS	DELFLDS	ERROR	CSIS
RELATION-SET MAPPING EXISTS FOR RELATION <RC-NAME>	DRPREL	ERROR	RELATION
RELATION-SET MAPPING EXISTS FOR RELATION <RC-NAME>	DRPRCE	ERROR	CSIS
RELATIONAL DATA BASE FIELDS MUST BE 01 LEVEL	RECSEMS	ERROR	RECORD
RELATIONAL DATA BASE FIELDS MUST BE 01 LEVEL	RECSEMS	ERROR	DATAFIELD
RELATIONAL DATA BASES DO NOT NEED FILLER INFORMATION - EXCLUDE THE COLUMN DEFINITION	RECSEMS	ERROR	RECORD

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
RELATIONAL DATA BASES DO NOT NEED KNOWLEDGE OF INDEXES OR KEYS - SEE FIELD <DF-ID>	RECSEMS	ERROR	RECORD
RELATIONAL DATA BASES DO NOT NEED KNOWLEDGE OF INDEXES /KEYS. SEE FIELD - <TDFT-DFID(TDFT-INDEX)>	FLDSEMS	ERROR	RECORD
RELATIONAL DATA BASES DO NOT SUPPORT REPEATING FIELDS NOT COBOL TYPE INDEXES NOR REDEFINITIONS	FLDSEMS	ERROR	RECORD
RELATIONAL DATA BASES DO NOT SUPPORT REPEATING FIELDS NOR COBOL TYPE INDEXES NOR REDEFINITIONS. SEE FIELD - <DF-ID>	RECSEMS	ERROR	RECORD
RELATIONAL DATA BASES MUST BE KNOWN TO CDM	RECSEMS	ERROR	RECORD
RELATIONS ASSOCIATED WITH ENTITY <ec-name> CAN NOT BE DROPPED	DRPENT	ERROR	ENTITY
RENAME-LIST TABLE EXCEEDED	MKRNLST	ERROR	RELATION
REPEATING FIELD OR REPEATING GROUP CANNOT BE KEY SEE FIELD - <TDFT-DFID (TDFT-INDEX)>	FLDSEMS	ERROR	RECORD
REPEATING FIELD OR REPEATING GROUP CANNOT BE KEY. SEE FIELD - <DF-ID>	RECSEMS	ERROR	RECORD

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
REUSABLE NUMBER NOT ADDED TO DATA BASE	ADDRNUM	FATAL	DATABASE
RIGHT DEPENDENT CARDINALITY <VALUE-X> TOO LARGE - VALUE UNCHANGED	ALTCARD	WARNING	
RIGHT DEPENDENT CARDINALITY <value-x> TOO LARGE - TRUNCATED TO 99	CHKCARD	WARNING	RELATION
RMVPAG ERROR IN PROCMD- RCODE IS <RCODE>	PROCMD	ERROR	
RPLFRM ERROR IN PROCMD - RCODE IS <RCODE>	PROCMD	ERROR	
RT-NO NOT FOUND FOR <RT-ID>	DBLOOK	ERROR	CSIS
RTP TABLE OVERFLOW ERROR	SELRTPM	ERROR	
RTP TABLE OVERFLOW ERROR	SELRTPM	ERROR	TABLE
SCHEMA NAMES COULD NOT BE INSERTED	DEFCODL	ERROR	DATABASE
SEC-DECOMPOSITION-LIST OVERFLOW	VERRELS	ERROR	VIEW
SEC/RC ENTITY COULD NOT BE CREATED FOR RELATION <VRCL-RC-NAME>	BLSECRC	ERROR	RELATION

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
SET (SET-NAME) NOT FOUND	CPYSET	ERROR	
SET <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	SET
SET <SET NAME> DOES NOT EXIST	MAPASET	ERROR	SET
SET <SET-ID> DOES NOT EXIST	DRPSET	ERROR	SET
SET <SET-ID> DOES NOT EXIST	VEROBJ	ERROR	SET
SET <SET-ID> IS USED IN SOFTWARE MOD.	SELRSET	ERROR	SET
SET <SET-ID> IS USED IN SOFTWARE MOD.	DRPSET	ERROR	SET
SET <SET-ID> IS USED IN SOFTWARE MOD.	SELSTM	ERROR	SET
SET <SET-ID> IS USED IN SOFTWARE MOD.	DELDBST	ERROR	SET
SET <SET-ID> MAPS TO A RELATION, NOT DELETED	DRPSET	ERROR	CSIS
SET <SET-ID> MAPS TO A TAG, NOT DELETED	DRPSET	ERROR	CSIS
SET <SET-ID> NOT DELETED FROM DESC-TEXT	DRPSET	ERROR	SET
SET <SET-ID> NOT DELETED FROM DF-SET-LINKAGE	DRPSET	ERROR	SET
SET <SET-ID> NOT DELETED FROM RECORD-SET	DRPSET	ERROR	SET
SET <SET-ID> NOT DELETED FROM SET-TYPE-MEMBER	DRPSET	ERROR	SET
SET <SET-NAME> DOES NOT EXIST	DRPSMAP	ERROR	SET

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
SET <SET-NAME> IS A MULTI MEMBER SET	MAPASET	ERROR	SET
SET MAP REQUIRES TWO SETS FROM SAME RECORD FOR EVERY TAG	ADDMAP	ERROR	CSIS
SET NAME <set name> ALREADY MAPS TO A RELATION CLASS	MAPASET	ERROR	SET
SET NOT DELETED FROM RECORD-SET	DELDBST	ERROR	
SET OF RECORD <RT-ID> NOT DELETED, MAPPING FOUND	DRPREC	ERROR	CSIS
SET OUTPUT TO FILE/SCREEN HAS NOT BEEN ESTABLISHED	CHKMOD	ERROR	MODEL
SET OUTPUT TO FILE/SCREEN HAS NOT BEEN ESTABLISHED	CPYVIEW	ERROR	VIEW
SETS FOR <DB-ID> NOT DELETED FROM RECORD-SET	DRPDB	ERROR	DATABASE
SETS FOR <DB-NAME> NOT DELETED FROM RECORD SET	DRPDB	ERROR	SET
SOFTWARE MODULE CURRENLTY EXISTS-CAN'T ADD	DEFSMOD	ERROR	MODULE
SOFTWARE MODULE DOESN'T EXIST	DRPSMOD	ERROR	MODULE
SOFTWARE MODULE DOESN'T EXIST -CAN'T ALTER	ALTSMOD	ERROR	MODULE
SOFTWARE MODULE IS BEING USED	DRPSMOD	ERROR	MODULE
SQL ERROR <MOD-NAME>	INSFLD	ERROR	RECORD
STANDARD DATA TYPE NAME NOT FOUND FOR DOMAIN:	ALLVIEW	ERROR	DOMAIN

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
STND. DATA TYPE COULD NOT BE RETRIEVED	ALTVLRG	ERROR	DOMAIN
TABLE HAS OVERFLOWED NO FURTHER FIELDS WILL BE CHECKED	ALTFLDS	ERROR	DATAFIELD
TABLE OVERFLOW ON EC-LIST	CHKOWN	ERROR	MODEL
TABLE OVERFLOW ON RC-DEPKC-LIST: INCREASE SIZE	ADDRCEC	ERROR	MODEL
TAG <TAG-NAME> ALREADY MAPPED TO DATABASE <DB-NAME> RECORD <REC-NAME>	MAPADF	ERROR	TAG
TAG <TAG-NAME> DOES NOT EXIST IN THE ENTITY	KCMADD	ERROR	TAG
TAG <TAG-NAME> FOR ENTITY <EC-NAME> HAS NO DOMAIN	CHKATT	WARNING	ATTRIBUTE
TAG <TAG-NAME> IS ALREADY A MEMBER OF THE KEY	KCMADD	ERROR	TAG
TAG MUST BE MAPPED TO AT LEAST TWO SETS FOR DATABASE	ADDMAP	ERROR	CSIS
TAG MUST BE MAPPED TO AT LEAST TWO SETS FOR DATABASE <DB-NAME>	DRPSMAP	ERROR	CSIS
TAG NAME <TAGNAME> AND PREF NO <PREFNO> DO NOT EXIST	ALTMAP	ERROR	CSIS
TAG NAME <tag name> DOES NOT EXIST	CRTMAP	ERROR	TAG
TAG NAME <tag name> DOES NOT EXIST	DRPMAP	ERROR	TAG
TAG NAME <tag name> DOES NOT EXIST	ALTMAP	ERROR	ATTRIBUTE
TAG NOT DELETED FOR <NEW-TAG-NO>	DELMIGK	ERROR	TAG
TAG NOT DELETED FOR <NEW-TAG-NO>	DMIGKRC	ERROR	TAG
TAG NOT DELETED FOR <NEW-TAG-NO>	DLMIGRC	ERROR	TAG

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
TAG NOT DELETED FOR <NEW-TAG-NO>	DELMIGRC	ERROR	TAG
TAG NOT MAPPED FOR ANY PREFERENCE	ALTSMAP	ERROR	TAG
TAG-NAME <NEW-TAG-NAME> IS A DUPLICATE NAME	ADDMIG	ERROR	TAG
TAGTBL OVERFLOWED	SAVTAGS	ERROR	CSIS/CSES
TDFT-TABLE OVERFLOW	RETRFLD	ERROR	RECORD
TEXT DESCRIPTIONS NOT DELETED FOR RC-NO <RC-NAME>	DRPRCE	ERROR	RELATION
TEXT DESCRIPTIONS NOT DELETED FOR RC-NO <RC-NAME>	DRPRCE	ERROR	RELATION
TEXTUAL DESCRIPTION FOR TAG NO NOT DELETED	DELAUC	ERROR	TAG
TEXTUAL DESCRIPTION NOT DELETED FOR <DF-ID>	DELFLDS	ERROR	DATAFIELD
TEXTUAL DESCRIPTION NOT DELETED FOR <DF-ID>	DELDBDF	ERROR	DATAFIELD
TEXTUAL DESCRIPTION NOT DELETED FOR <SET-ID>	DELFLDS	ERROR	DATAFIELD
TEXTUAL DESCRIPTION NOT DELETED FOR <SET-ID>	DELDBST	ERROR	
TEXTUAL DESCRIPTIONS NOT DELETED FOR <DOM-NAME>	DRPDOM	ERROR	DOMAIN
THE ENTITY <EC-NAME> DOES NOT MAP TO ANY ATTRIBUTES	SELTGEC	ERROR	CSIS
THERE EXISTS A RELATION CLASS TO SET MAPPING	DRPDB	ERROR	DATABASE
THERE EXISTS AN AUC TO INTERNAL SCHEMA MAP	DRPDB	ERROR	DATABASE
THERE EXISTS AN UNION MAPPING	DRPDB	ERROR	DATABASE
TO ENTITY NAME MUST BE ENTERED	COPENT	ERROR	ENTITY

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
TO-ENTITY <ENT-NAME> DOES NOT EXIST	CMBENT	ERROR	ENTITY
TO-ENTITY NAME <TO-EC-NAME> ALREADY EXISTS	COPENT	ERROR	ENTITY
TO-ENTITY NAME AND FROM-ENTITY NAME ARE THE SAME	COPENT	ERROR	ENTITY
TOTAL DATA BASE COULD NOT BE INSERTED	DEFTOT	ERROR	DATABASE
TRYING TO GET APNAME	CRTVMOD	ERROR	DOMAIN
TRYING TO GET OUTPUT FILE NAME	CRTMACR	ERROR	
TRYING TO GET OUTPUT FILE NAME	CRTVMOD	ERROR	DOMAIN
TYPE FOR ALTERING STD. DATA TYPE NOT = S, I O R C	ALTDT	ERROR	USERTYPE
TYPE FOR STD. DATA TYPE NOT = N, S OR C	ADDSTD	ERROR	USERTYPE
TYPE IS NOT LEGAL - <TYPE-ID-TEMP>	PROCDT	ERROR	
TRYING TO GET APNAME	CRTMACR	ERROR	
UNABLE TO ADD FIELDS	ALTREC	ERROR	RECORD
UNABLE TO ALTER RECORD. ERROR OCCURRED DURING INSERTION <...>	ADDFLDS	ERROR	RECORD
UNABLE TO ASSIGN NEW NUMBER	GETNNUM	FATAL	DATABASE
UNABLE TO ASSIGN UNIQUE KC NUMBER	TERKEY	ERROR	KEY
UNABLE TO CREATE DATA ITEM NUMBER FOR <DI-ID>	INSDI	ERROR	DATAITEM
UNABLE TO CREATE USDF-DT-NO FOR <DATA-TYPE-NAME>	INSDT	ERROR	USERTYPE
UNABLE TO DELETE COMPLETE RELATION FOR KEY CLASS <KC-NAME>	DRPKC	ERROR	KEY

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
UNABLE TO DELETE INDEX FIELD	ALTFLDS	ERROR	DATAFIELD
UNABLE TO DELETE KEY CLASS MEMBERS FOR: <KC-NAME>	DRPKC	ERROR	KEY
UNABLE TO DELETE MIGRATING KEY CLASS MEMBER	DRPMGKM	ERROR	KEY
UNABLE TO DELETE MIGRATING KEY CLASS MEMBER	DRPMGRC	ERROR	KEY
UNABLE TO DELETE OLD RECORD	ADDFLDS	ERROR	RECORD
UNABLE TO DROP AREA <AREA-ID>	ALTREC	ERROR	RECORD
UNABLE TO GET DATA BASE ID	ALTFLDS	ERROR	DATAFIELD
UNABLE TO GET THE DATA BASE ID	DEFREC	ERROR	RECORD
UNABLE TO GET THE DATA BASE ID	ALTREC	ERROR	RECORD
UNABLE TO INSERT <CDFT-DFID>	ALTFLDS	ERROR	DATAFIELD
UNABLE TO INSERT AC KEYWORD	ADDKWA	ERROR	KEYWORD
UNABLE TO INSERT COMPLETE RELATION	ADDMIG	ERROR	RELATION
UNABLE TO INSERT DATA_ITEM: <TAG-NAME>	ALLVIEW	ERROR	DATAITEM
UNABLE TO INSERT DATA_ITEM: <TAG-NAME>	SMVIEW	ERROR	DATAITEM
UNABLE TO INSERT EC KEYWORD	ADDKWE	ERROR	KEYWORD
UNABLE TO INSERT KEY CLASS <KC-NAME> INTO MODEL	ADDKC	ERROR	KEY
UNABLE TO INSERT KEYWORD <key-word>	ADDKW	ERROR	KEYWORD
UNABLE TO INSERT PROJECT_DATA_ITEM: <TAG-NAME>	SMVIEW	ERROR	DATAITEM
UNABLE TO INSERT PROJECT_DATA_ITEM_ <TAG-NAME>	ALLVIEW	ERROR	DATAITEM

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
UNABLE TO INSERT RC KEYWORD	ADDKWR	ERROR	KEYWORD
UNABLE TO INSERT RECORD <RT-ID>	DEFFLDS	ERROR	RECORD
UNABLE TO INSERT RECORD SET: <SET-ID>	DEFSET	ERROR	RECORD
UNABLE TO INSERT RECORD SET: <SET-ID>	DEFSET	ERROR	RECORD
UNABLE TO INSERT RELATION NAMED <RC-NAME> INTO MODEL	CRTREL	ERROR	RELATION
UNABLE TO INSERT SET TYPE MEMBER: <MEMBER-ID>	DEFSET	ERROR	DATAFIELD
UNABLE TO INSERT SET TYPE MEMBER: <MEMBER-ID>	DEFSET	ERROR	DATAFIELD
UNABLE TO INSERT TAG NAME <NEW-TAG-NAME> AS ATTRIBUTE -USE-CLASS	ADDMIG	ERROR	TAG
UNABLE TO INSERT TAG NAME <NEW-TAG-NAME> AS INHERITED-ATTRIBUTE-USE-CLASS	ADDMIG	ERROR	TAG
UNABLE TO MODIFY <CDFT-DFID>	ALTFLDS	ERROR	DATAFIELD
UNABLE TO OPEN FILE	TRAFIL	ERROR	DATABASE
UNABLE TO RETRIEVE DB SET FOR <DB>	DEFSET	ERROR	DATABASE
UNABLE TO RETRIEVE DB SET FOR <DB>	DEFSET	ERROR	DATABASE
UNABLE TO RETRIEVE RECORD NUMBER FOR <RT-ID>	GENMREC	ERROR	RECORD
UNABLE TO RETRIEVE RECORD VALUES	ADDFLDS	ERROR	RECORD
UNABLE TO UPDATE ATT CL <attribute class number>	CHGDOM	ERROR	ATTRIBUTE
UNION MAPPING EXISTS FOR ENTITY <EC-NAME>	DRPENT	ERROR	ENTITY

ERROR MESSAGE	ROUTINE	ERR TYPE	OBJECT TYPE
UNIQUE OBJECT NUMBER CANNOT BE ASSIGNED	GETNNUM	FATAL	DATABASE
UNSUCCESSFUL RETRIEVAL OF RECORD <RT-ID>	ALTFLDS	ERROR	DATAFIELD
USER SPECIFIED DB_NAME.SET_NAME NOT VALID FOR THIS TAG NAME	ALTSMAP	ERROR	SET
USERDATATYPE <UDFDT-NAME> DOES NOT EXIST	VEROBJ	ERROR	USERTYPE
USERDATATYPE <UDFDT_NAME> DOES NOT EXIST	VEROBJ1	ERROR	USERTYPE
USER_VIEW COULD NOT BE DELETED FOR : <VIEW-NAME>	DRPVIEW	ERROR	VIEW
UT TABLE OVERFLOW ERROR	GENEUN	ERROR	CSIS
VALIDATION OF DATA TYPE FAILED	CHKDOMS	ERROR	USERTYPE
VALIDATION OF STANDARD DATA TYPE FAILED	CHKDOMS	ERROR	USERTYPE
VALIDATION OF STD DATA TYPE FAILED	ALLVIEW	ERROR	USERTYPE
VALIDATION OF STD DATA TYPE FAILED	SMVIEW	ERROR	DOMAIN
VALUE COULDN'T BE UPDATED= <DOM-VALUE>	ALTVLRG	ERROR	DOMAIN
VALUE DOES NOT MATCH DEF. <DOM-VALUE>	CRTVLRG	ERROR	DOMAIN
VALUE TO BE DROPPED NOT FOUND <DOM-VALUE>	ALTVLRG	WARNING	
VALUES NOT INSERTED INTO DATA_ITEM FOR: <DI-NAME>	CHKDOMS	ERROR	DATAITEM
VALUES NOT INSERTED INTO PROJECT-DATA-ITEM DI-NAME: <DI-NAME>	CHKDOMS	ERROR	DATAITEM
VALUES NOT INSERTED INTO PROJECT-DATA-ITEM FOR: <DI-NAME>	CHKDOMS	ERROR	DATAITEM
VERIF. MOD. GENERATED = <CRT-APNAME> ON FILE: <VMODFILE>	CRTVMOD	WARNING	DOMAIN

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
VERIF. MOD. GENERATED=<MOD-APNAME> ON FILE :<VMODFILE>	ALTVMOD	WARNING	
VIEW <OBJ-ID-1> DOES NOT EXIST	VEROBJ	ERROR	VIEW
VIEW <OBJ_ID_1> DOES NOT EXIST	VEROBJ1	ERROR	VIEW
VIEW <VIEW-ID> DOES NOT EXIST	CPYVIEW	ERROR	VIEW
VIEW <view-name> COULD NOT BE CREATED	CRTVIEW	ERROR	VIEW
VIEW ALREADY EXIST : <view-name>	CRTVIEW	ERROR	VIEW
VIEW COULD NOT BE CREATED : <view-name>	CRTVIEW	ERROR	VIEW
VIEW DOES NOT EXIST FOR : <VIEW-NAME>	DRPVIEW	ERROR	VIEW
VIEW PARTICIPATES IN ALGORITHM/PRECOMPILATION	DRPVIEW	ERROR	VIEW
VIEW USED DATAITEM <DI_ID> DOES NOT EXIST	VEROBJ1	ERROR	DATAITEM
VIEW-DATA-ITEM-LIST TABLE OVERFLOWED	BLVWLST	ERROR	VIEW
VIEW-EC-XREF COULD NOT BE DELETED FOR: <VIEW-NAME>	DRPVIEW	ERROR	VIEW
VIEW-FROM-LIST TABLE OVERFLOWED	BLVWLST	ERROR	VIEW
VIEW_QUALIFY_CRITERIA NOT DELETED FOR: <VIEW-NAME>	DRPVIEW	ERROR	VIEW

ERROR MESSAGE	ROUTINE	ERR TYP	OBJECT TYPE
VIEW_QUAL_XREF NOT DELETED FOR: <VIEW-NAME>	DRPVIEW	ERROR	VIEW
WARNING: ATTRIBUTE MIGRATIONS ARE BEING DELETED; THIS WILL RESULT IN SUSPECT KEY CLASSES.	PRESMIG	WARNING	ATTRIBUTE
WARNING: DESCRIPTION TYPE <DESC_TYPE> DOES NOT EXIST	CPYDSTP	WARNING	DESC_TYPE
WARNING: ENTITY <ENTITY NAME> DOES NOT EXIST IN THE FROM MODEL	CPYMOD	WARNING	ENTITY
WARNING:ATTRIBUTE MIGRATES ARE BEING DELETED; THIS WILL RESULT IN SUSPECT KEY CLASSES.	AOADPM	WARNING	ATTRIBUTE
WARNING:ENTITY <EC-NAME> HAS NOT ATTRIBUTE-USE CLASS	CHKATT	WARNING	ENTITY
WARNING:NONKEY CANNOT BE SPECIFIED AFTER FIRST TIME	AOAEVAL	WARNING	KEY
WHERE CLAUSE MUST BE SPECIFIED IN MULTIPLE TABLE VIEW	BLVWLST	ERROR	VIEW
YOU MAY NOT OWN ATTRIBUTE NAME <AC-NAME>	ADDENT	WARNING	ATTRIBUTE
YOU MAY NOT OWN ATTRIBUTE NAME: <OWNED-AC-NAME>	ADDATT	ERROR	ATTRIBUTE

1203 records selected.

APPENDIX B

GLOSSARY

Alpha-Numeric Data Format

A data format for values that can contain characters other than numerals (0-9). Numerals may be permitted also.

Attribute Class

A collection of all the same kind of attributes; i.e., those that have the same meaning. An attribute is a characteristic or fact about an entity. An attribute consists of a name (e.g., employee hire date) and a value (e.g., 15 August 1980). An attribute value may be:

A. Nondivisible (e.g., state name)

B. Divisible, such as a concatenation of two or more other attribute values (e.g., part number formed by concatenating drawing number and material code).

C. Computed from one or more other attribute values (e.g., age computed as current date minus birth date).

Attribute Class Data Description

A generic data description that applies to a particular attribute class.

Attribute Use Class

A model attribute class that appears in a model entity class. Each attribute use class represents either an owned attribute class or an inherited attribute class.

Attribute Use Class/Data Item Mapping

Indicates that an attribute use class corresponds to a data item i.e., that they have the same meaning and that the data item can be used to access the values for the attribute use class.

Attribute Use Class/Record Set Mapping

Certain attribute use classes can be represented in a database by a group of record sets rather than by a data field. For example, Project could be represented by Task record sets called Pending, In-Process, On-Hold, and Completed. An attribute use class/record set mapping indicates that a particular record set corresponds to a particular attribute use class value.

Component Data Field

A data field that is part of another data field; if data field A is made up of data fields B, C, and D, each of these latter data fields is a component of A. A data field cannot be a component of more than one other data field.

Component Domain

An elementary domain that is part of another domain; a Date domain might be made up of a Month domain, a Day of Month domain, and a Year domain. Each of these latter domains would be a component of the Date domain. An elementary domain can be a component of several other domains.

DBD

An IMS database is defined by a Database Description (DBD). The DBD consists of statements which map an IMS structure into physical storage.

Database Area

A subdivision of a CODASYL database. This subdivision is a technique for improving the efficiency when accessing database record type instances. When a database is subdivided into database areas, some or all of its records types are assigned to particular areas. Instances of these record types are stored only within the assigned areas. Then, these record type instances can be accessed by searching only the appropriate areas rather than the entire database. This access method is only used when the record type instances cannot be located by other means (e.g., by calc keys or record sets).

Database Area Assignment

Indicates that a record type is assigned to a database area.

Database Directory

A software library that must be used when accessing a database.

Database Password

A code that must be supplied when logging on to a DBMS to use a database. The DBMS verifies the password before accepting any other messages.

Data Field

A portion of a record type in which data values can be stored.

Data Field/Record Set Linkage

A data field in a variable data set in a TOTAL database that is used as the variable control key for a linkpath from a master data set.

Data Field Redefinition

A data field that occupies the same space in a record type as another data field. A record instance cannot contain values in both data fields. One instance can contain a value in one field while another contains a value in the other.

Data Item

An attribute class as seen by a user in a user view; i.e., a kind of data (e.g., employee hire date) not a particular data value (e.g., 15 August 1980).

Data Type

The combination of a type of values (e.g., alphanumeric, signed numeric, etc.) and a type of storage (e.g., binary, packed, etc.)

Data Type Name

Names of NDDL data types. The NDDL data types correspond to the following COBOL/FORTRAN data types:

NDDL Data Type	COBOL/FORTRAN Data Type
INTEGER	FORTRAN binary integer
CHARACTER	x(n)
SIGNED	S99V99
FLOAT	FORTRAN floating point
UNSIGNED	99V99
PACKED	COMP-3

Description Type

A generic object may have several different kinds or styles of description (short, long, technical, nontechnical, etc.). Each is a description type.

Domain

A set of rules about the values that are allowed for a data item, attribute class, or data field. A domain is either an elementary domain or a group of two or more elementary domains, called component domains.

Domain Range

A series of consecutive values that represent all or part of an elementary domain.

Domain Value

A single value within an elementary domain.

Elementary Data Field

A data field that does not have any component data fields.

Entity Class

A collection of similar entities; i.e., those that have the same kinds of attributes. An entity is a person, place, event, thing, concept, etc.

Entity Class/Record Type Mapping

Indicates that an entity class corresponds to a record type; i.e., that they both have the same meaning and that the record type can be used to store instances of the entity class.

If a record type has more than one EC-RT mapping, some of its instances correspond to instances of one entity class while others correspond to instances of another; i.e., the record type is the relational union of the entity classes. An example is a Replenishment Order record type that maps to both the Purchase Order and Manufacturing Order entity classes. Each record instance represents either a purchase order or a manufacturing order.

Feedback

The length of the key feedback area for an IMS PCB. When IMS retrieved a segment from the database, the requested segment is fetched and a fully concatenated key is placed in the key feedback area. The fully concatenated key consists of the concatenation of the sequence field of values of all segments in the hierarchical path from the root down to the retrieved segment. The key feedback area must be large enough to accommodate the maximum length for a fully concatenated key and stated in the KEYLEN entry of the PCB macro.

File

A set of stored data that is managed by a file management] system (e.g., VSAM).

File/Database

A set of stored data; i.e., either a computer file (e.g., a VSAM or flat file) or a database (e.g., an ORACLE or IMS database).

Generic Data Description

A detailed description of the values for one or more data items, attribute classes, data fields, and/or module parameter. It includes format, measurement, and domain characteristics of the values.

Generic Data Description Domain

A domain that is specified as part of a generic data description.

Generic Data Description Unit of Measure

A unit of measure that is specified as part of a generic data description.

Host

A computer in the IISS.

IMS Segment

A record type in a database that is controlled by IBM's IMS DBMS.

Inherited Attribute Class

A key class in the independent entity class of a relation class that has migrated to appear in the dependent entity class of that relation class.

Key Class

A group of one or more of an entity's attributes that can be used to uniquely identify the entity within its entity class. An entity can have more than one key. A key class is a collection of the attribute classes whose member attributes comprise the keys for the entities in an entity class. An entity class has the same number of key classes as each of its member entities has keys. For example, if each entity has three keys, the entity class has three key classes.

Key Class Member

An attribute use class that is part of a key class.

Keyword

A word that has been designated as a means of locating a generic object or a number of similar generic objects.

Model

A representation of the information requirements of all or part of an enterprise in terms of entity classes, relation classes, and attribute classes.

Object Type

Sets of attributes are, in relational terms, called objects. Objects participate in relationships with other objects. Entities within the Common Data Model (see Generic Object in the CDM1 Doc. Control No. CCS620141000) are called OBJECT TYPES for the Integrated Information Support System.

Owned Attribute Class

A model attribute class that appears as an attribute use class in a model entity class and is not an inherited attribute class.

Program Control Block

A portion of a PSB that describes and controls how an IMS database can be accessed.

Program Specification Block

A group of logical views of IMS databases that is used for interacting with the IMS DBMS.

Record Set

An association between one record type, called the owner, and one or more other record types, called the members.

Record Set Member

A record type that is a member of a record set.

Record Type

A group of data values that are stored together as a unit in a computer file or database. A record type is the collection of all the records of the same kind; i.e., all the records that contain the same kind of data values.

Relation Class

An association between an entity in one entity class and one in another. A relationship has a label that describes the association. For example, a customer named ABC Corp. is associated with an order numbered 123 in a manner labeled "placed". A relation class is a collection of the identically labeled relationships between the members of the same two entity classes. Each relation class is either specific or non-specific.

In a specific relation class, one entity class is "independent" while the other is "dependent"; entities in the first can exist without being associated with any in the second, but those in the second cannot exist without being associated with one in the first. One key class from the

independent entity class "migrates" through each specific relation class to appear in the dependent entity class as inherited attribute classes.

In a non-specific relation class, neither entity class is dependent on the other; entities in either entity class can exist without being associated with any in the other. For convenience, one entity class is arbitrarily called "independent" and the other is called "dependent".

Segment Data Field

A data field that is an IMS segment.

Subschema

The description, in the DDL of a CODASYL DBMS, of all or part of a database. For IISS, only one subschema is needed for a CODASYL database and it must describe all the common data within the database that is to be accessible with NDML.

Tag Name

A unique name for an attribute use class within an entity class.

Unit of Measure

A standard scale for determining the magnitude of something. Examples include inch, foot, foot-inch, meter, ounce, pound, hour, minute, second, etc.

User View

A group of data items that a user wants to deal with as a group. It is similar to an entity class but does not necessarily meet all the conditions for being one; it can be thought of as an unnormalized entity class. A user view is often the result of combining several entity classes via relational join operations and selecting particular attribute use classes as data items via relational project operations.

APPENDIX C

REFERENCES

Related ICAM Documents included:

UM620341001	CDM Administrator's Manual
TBM620341000	CDM1:IDEF1 Model of the CDM
PRM620341200	NDML Programmer's Reference Manual
UM620341002	Information Modeling Manual-IDEF1 Extended
DS620341200	NDML Precompiler Development Specification
DS620341320	Data Aggregators Development Specification
DS620341310	Distributed Request Supervisor Development Specification